Increasing the Precision of Forest Area Estimates through Improved Sampling for Nearest Neighbor Satellite Image Classification

Christine E. Blinn

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Forestry

Randolph H. Wynne, Chair
Richard G. Oderwald
Stephen P. Prisley
Gregory A. Reams
John A. Scrivani

July 29, 2005
Blacksburg, Virginia

Keywords: training data, band combinations, Landsat ETM+, nearest neighbor, forest area estimation

Increasing the Precision of Forest Area Estimates through Improved Sampling for Nearest Neighbor Satellite Image Classification

Christine E. Blinn

(ABSTRACT)

The impacts of training data sample size and sampling method on the accuracy of forest/nonforest classifications of three mosaicked Landsat ETM+ images with the nearest neighbor decision rule were explored. Large training data pools of single pixels were used in simulations to create samples with three sampling methods (random, stratified random, and systematic) and eight sample sizes (25, 50, 75, 100, 200, 300, 400, and 500). Two forest area estimation techniques were used to estimate the proportion of forest in each image and to calculate forest area precision estimates. Training data editing was explored to remove problem pixels from the training data pools. All possible band combinations of the six non-thermal ETM+ bands were evaluated for every sample draw. Comparisons were made between classification accuracies to determine if all six bands were needed. The utility of separability indices, minimum and average Euclidian distances, and cross-validation accuracies for the selection of band combinations, prediction of classification accuracies, and assessment of sample quality were determined.

Larger training data sample sizes produced classifications with higher average accuracies and lower variability. All three sampling methods had similar performance. Training data editing improved the average classification accuracies by a minimum of 5.45%, 5.31%, and 3.47%, respectively, for the three images. Band combinations with fewer than all six bands almost always produced the maximum classification accuracy for a single sample draw. The number of bands and combination of bands, which maximized classification accuracy, was

dependent on the characteristics of the individual training data sample draw, the image, sample size, and, to a lesser extent, the sampling method. All three band selection measures were unable to select band combinations that produced higher accuracies on average than all six bands. Cross-validation accuracies with sample size 500 had high correlations with classification accuracies, and provided an indication of sample quality.

Collection of a high quality training data sample is key to the performance of the nearest neighbor classifier. Larger samples are necessary to guarantee classifier performance and the utility of cross-validation accuracies. Further research is needed to identify the characteristics of "good" training data samples.

ACKNOWLEDGEMENTS

I would like to thank the members of my committee, Randolph H. Wynne, Richard G. Oderwald, Stephen P. Prisley, Gregory A. Reams, and John A. Scrivani, for their support and intellectual contributions to each stage of my research efforts. I am especially thankful to John Scrivani and the Virginia Department of Forestry for their willingness to share data and resources in support of collaborative research efforts. I would also like to thank the USDA Forest Service, FIA Program, especially the Southern and North Central Research Stations, for their financial support of my research. Funding was also provided by NASA (NAG5-10548 and NNS-04AB26G) and the US McIntire-Stennis Research Program (VA-136589). I am extremely appreciative of all the support I received.

I have greatly enjoyed my intellectual and personal interactions with my fellow graduate students and colleagues, especially Beyhan Amichev, Zack Bortolot, Katie Joseph, Rebecca Musy, Sorin Popescu, and Jan Van Aardt. I would also like to thank Memo Belgin for his help with my learning to use Ojibwa and Rhonda Phillips for her essential assistance with programming, specifically the Quick Sort code.

I could not have asked for a more energetic, supportive, and intelligent advisor than I found in Randy Wynne. His willingness to provide any assistance necessary to help his students learn and accomplish their goals goes far beyond all expectations. I am honored to have had the opportunity to work closely with him.

Last but not least, I am grateful to my family and friends for their patience and support through my many years of higher education. My parents, Raymond and Patricia Blinn, have been a continued source of love and understanding through out my life and a safety net I can always count on. My grandparents, Kenneth and Mary LaTourette, have also been very supportive of my educational goals. My significant other, Tommy Regan, and his family, especially Susan and Edwin Keith, and the late Lydia Langdon, have provided me with love, companionship, and nourishment. Lastly, my best friend, Autumn Birt, and her parents have been a continued source of encouragement. To all I am extremely grateful for their roles in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1
INTRODUCTION

1.1 Forest Area Estimation

Forest area estimation is an important phase, or requirement, in many monitoring and modeling efforts. Whether a project sets out to monitor the change in forest area, calculate the carbon budget of a forest system, or estimate the total volume of timber in a forest tract, the total area of forest in the region of interest must be determined. When the scale of a project is at a state, national or global level, it is not feasible to determine the forest area from ground measurements alone. Remote sensing becomes the most viable option for creating forest area estimates over large areas (Bauer 1976).

The USDA Forest Service (FS) has been tasked with monitoring the current extent and condition of the forest resources in the United States, and accomplishes this task through the Forest Inventory and Analysis (FIA) Program. A goal of the first phase (Phase 1) of the FIA program is to estimate the area of forest in each state or region. One method that has been used to calculate forest area estimates is based on double sampling. In the first stage of double sampling, points on a systematic hexagonal grid (Figure 1), which will be referred to as Phase 1 points, are classified as forest or nonforest. During the second stage of double sampling, a subset of the Phase 1 points are visited on the ground to verify their classification. The proportion of Phase 1 points in each category are calculated and adjusted based on the ground truth data to obtain phase 1 forest area estimates. During a second phase of sampling (Phase 2), forest attribute data is collected on permanent ground plots, which will be referred to as FIA ground plots. These plots have random locations within each hexagon of a second hexagonal grid (Figure 1). Forest population attributes are estimated by combining the area estimates from Phase 1 with the FIA ground plot data.

Previously, FIA phase 1 points were photo-interpreted over high resolution aerial photography. The recent change to an annual inventory system, where one fifth of all FIA ground plots are measured each year, resulted in the need for more rapid area estimates (Reams and Van Deusen 1999). Reams and Van Deusen (1999) stated that the old method of photo-interpretation (PI) could take up to one year to complete. Additionally, the availability and cost of obtaining higher resolution imagery on a more frequent time scale are also limiting factors to the continued use of the PI method for forest area estimation (Wayman et al. 2001). Satellite image classifications have started to replace the PI method for obtaining large area forest area estimates. The tradeoffs between spatial, spectral, and radiometric resolution of different satellite platforms for this task have been discussed by Czaplewski (1999) and Wynne et al. (2000). Landsat TM imagery has been selected by several Research Stations within the USDA FS for use in obtaining phase 1 forest area estimates for the FIA Program (McRoberts and Hansen 1999, Hoppus et al. 2000, Rack 2000, Wayman et al. 2001, McRoberts et al. 2002a, 2002b, Musy et al. in press).

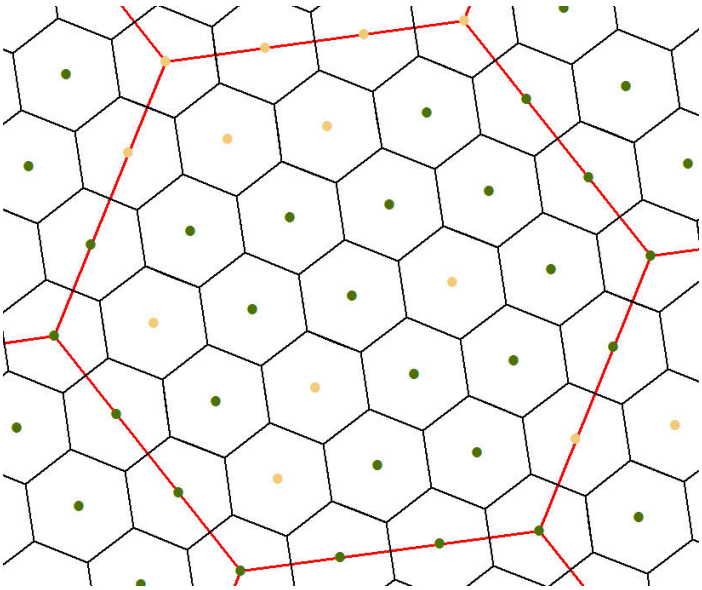
Area estimates obtained by counting the number of pixels in a classified image that were assigned to a given class (referred to as map marginals) and multiplying by the area of each pixel are known to be biased (Gallego 2004). In addition to the need to correct for the bias in map marginals, it is critical to also have a measure of the precision of the corrected area estimates. Once the map marginals are corrected and the precision of the corrected area estimate calculated, there is no longer a map, just an aggregate area estimate for a specified area. In the case of the FIA program, these estimates are used in the estimation of forest attributes like the total volume of hardwoods in a state or region. Clearly, minimizing the bias and improving the precision of

area estimates is very important. A number of techniques have been explored in the literature to correct the bias in map marginals and to calculate the precision of area estimates.

Three area estimation methods for use with satellite image classifications were investigated by Yuan (1997) for the correction of map marginals based on the classification's error matrix. Each of three estimators, direct (Card 1982), inverse (Bauer et al. 1978), and additive (Dymond 1992), had different feasible regions based on the sampling fraction and number of classes, but both the direct and inverse estimator were asymptotically unbiased based on Monte Carlo simulations (Yuan 1997). Czaplewski and Catts (1992) found the direct estimator, which they called the inverse estimator, to be superior to the inverse estimator, which they called the classical estimator. The direct estimator, or Card method, as it will be referred to herein, has been applied to FIA Phase 1 forest area estimation by Wayman et al. (2001) and Musy et al. (in press), as recommended by Reams and Van Deusen (1999) and Wynne et al. (2000). Another technique, that uses both satellite image classifications and FIA ground plots for forest area estimation, is based on the stratified estimation procedure from Cochran (1977). This procedure takes into account the proportion of forest observed at each of the FIA ground plots (McRoberts et al. 2002a).

Although other methods, including regression estimators (Deppe 1998), have been used for forest area estimation, only the Card and Cochran/McRoberts methods have commonly been applied to FIA Phase 1 forest area estimation, which is the focus of this study. Regardless of the method, the forest area precision estimate is directly related to the classification accuracy and the number of reference samples used to assess the accuracy (Musy et al. in press). Larger reference sample sizes and more accurate classifications produce more precise area estimates.

Figure 1. FIA Phase 1 Hexagonal Grid Overlaid with Phase 2 Hexagonal Grid



Each phase 1 hexagon, outlined in black, is approximately 90 hectares. Within each phase 1 hexagon is a phase 1 point color coded green for forest or tan for nonforest. Twenty seven phase 1 hexagons fit within each phase 2 hexagon, outlined in red, which is approximately 2428 hectares.

1.2 Training Data

Training data quality largely determines the resulting classification accuracy. Whether supervised, unsupervised, or hybrid image classification techniques are employed, training data is required at some phase of the classification process. Training data may be used to define the spectral characteristics of each informational class *a priori* in the case of supervised techniques, used subsequent to the creation of spectral classes (unsupervised techniques) to label the resulting spectral classes, or used at intermediate steps in a hybrid classification approach. Regardless of the stage at which training data is needed in the classification process, high quality training data is essential. However, there is not a simple, quantitative, non-subjective measure of the quality of a training data set that can be calculated prior to the classification process. The

quality and properties of a training data set are influenced by the sampling unit, sample size, and sampling method. Each of these factors is discussed in subsequent sections.

1.2.1 Sampling Unit

There are two basic sampling units for the collection of a training data sample: a single pixel; or a group of contiguous pixels. Obviously, a large variety of sizes and dimensions exist for a group of pixels depending on whether each is created from: a square or rectangular window; a polygon; an area of interest grown from a seed pixel; or an object from image segmentation. An issue with contiguous pixel training, that is frequently noted in the literature, is the existence of spatial autocorrelation, which results in pixels that are spatially closer to one another having a greater likelihood of being similar (Curran 1988, McGwire et al. 1993). Craig (1979), Campbell (1981), and Labovitz and Masuoka (1984) have shown that adjacent pixels in Landsat data are not independent and result in biased estimates of class covariance matrices with the tendency to underestimate class variability when used to create training class statistics. Training samples collected by single pixel training have performed as well, and in some cases better, than the traditional use of contiguous pixels (Campbell 1981, Gong and Howarth 1990, Chen and Stow 2002).

1.2.2 *Sample Size*

In terms of sample size, it has been recommended that at least 100 pixels be collected from several training areas for each informational class (Campbell 2002). Other sources recommend a sample size equal to between 10 and 100 times the number of bands (Lillesand et al. 2004), which results in the sample size rapidly increasing as data dimensionality increases. Rules of thumb are useful in selecting a training data sample size, but may result in the overestimation of the number of samples needed in order to err on the side of caution.

Several studies have reported increases in classification accuracy with increasing training data sample size (Maselli et al. 1992, Dobbertin and Biging 1996, Arora and Foody 1997, Pal and Mather 2003, Foody and Mathur 2004a). As can be seen in the comparison table (Table 1) of the studies listed in this section, a wide variety of imagery was used, with spatial resolutions between two and thirty meters and spectral resolutions between three and eleven bands. Given the array of parametric and nonparametric techniques used in these studies, it appears that sample size is important regardless of classification method. Yet, in at least one study training data sample size was only important with 1 of the 3 algorithms tested, neural networks (Arora and Foody 1997).

Chen and Stow (2002) noted an interaction between an appropriate sample size and the heterogeneity of the land cover in their study, where larger sample sizes were not always needed. They concluded that sample size was important with single pixel training. As their findings were based on the use of high resolution imagery, it is not clear if the same will be true for medium resolution satellite imagery. In another Foody and Mathur (2004b) paper, the ability to use a smaller sample size and still achieve the same accuracy as a sample four times its size was reported. This study used support vector machines (SVM), where training samples which help define the decision boundaries in feature space between classes appear to be more important than samples which capture the central tendencies of a class. The potential for smaller training data sample sizes to perform as well as larger sample sizes may be feasible if the important training sample characteristics can be identified.

Of the studies that indicated an increase in accuracy with increasing sample size, only two included forest in their classification scheme (Maselli et al. 1992, Dobbertin and Biging 1996) and only one of these used real imagery (Maselli et al. 1992). Although the Maselli et al.

(1992) study used Landsat TM data and created ten samples for each sample size, sample sizes smaller than 250 pixels were not examined. Similarly, Pal and Mather (2003) used Landsat ETM+ data, but only considered total sample sizes of 700 and larger. The only similar parameters between these studies were: (1) the almost universal use of single pixel training, and (2) the frequent use of stratified random sampling. Since a very small number of replicates for each training data sample size were used in these studies, the findings cannot be considered conclusive on the issue of sample size. A more exhaustive look at this relationship, over a larger study area and specifically for the purpose of forest area estimation, could lead to better insight into the tradeoffs between sample size and classification accuracy.

Interaction between the size of the training data set and the optimal number of bands has been suggested by Hughes (1968). Muasher and Landgrebe (1984) describe the "Hughes phenomenon" as referring to the fact that improvements in classification accuracy, resulting from increasing the number of bands, eventually level off instead of continuing to improve when a limited size training data set is used. The number of bands at which this leveling off occurs is related to the size of the training data set. Larger samples result in accuracies that do not level off until a larger number of bands are used. Based on these observations, the same or better classification accuracy can often be achieved by the use of fewer than the total number of available bands.

1.2.3 Sampling Method

The sampling method used to obtain a training data sample will influence the properties of the sample. Random, stratified random, systematic and cluster sampling are commonly used sampling methods (Cochran 1977). Simple random sampling has long been the preferred statistical choice for a sampling scheme due to the independence of samples and simplicity.

Although simple random samples do not place any limits on statistical inference, they may result in a nonrepresentative sample, especially with small sample sizes (Shiver and Borders 1996). Under-representation of spectral or informational classes that are rare in the landscape and/or that cover small areas are especially problematic with random sampling (Jensen 2005).

Since variation in spectral response can often occur across an image due to changes in illumination or atmospheric effects, such as haze, how well a training data sample is distributed spatially across an image may play an important role in determining the utility of a sample (Conese et al. 1993). Stratification is often used to assure the collection of samples in particular categories of interest, whether they are informational classes, spectral classes, geographic zones, or topographic zones.

Systematic sampling is one way of assuring complete spatial coverage of an entire image. The potential drawback of systematic sampling occurs when there is a systematic pattern of land use that is identical in spacing to the spacing of the sample. Systematic sampling has been recommended for use with data sets that are spatially autocorrelated. Webster et al. (1989) state that systematic sampling is most efficient for remote sensing applications due to the spatially correlated and largely stochastic nature of radiation. Curran (1988) also described the potential for large increases in precision with systematic sampling in contrast to simple random when the variance is spatially dependent. Yet, while for autocorrelated populations stratified random sampling is superior to simple random sampling, variations in landscape pattern make it impossible to state a general result with respect to systematic sampling (Cochran 1977).

Despite limited studies on appropriate sampling methods for training data collection, sampling methods have been explored in the context of accuracy assessment. The existence of spatial autocorrelation in the distribution of classification errors in a classified image (Congalton

1988) has led to numerous studies on an appropriate sampling method for accuracy assessment. Stehman (1999) states that systematic sampling is generally favored over simple random and cluster sampling due to autocorrelation in map errors. When used for accuracy assessment, systematic sampling designs were found to generally be more precise than simple random sampling due to the sample being spatially well-distributed (Stehman 1992, Stehman and Czaplewski 1998). Stratified sampling will result in precision gains over simple random sampling when the proportions of map errors in each stratum are significantly different (Cochran 1977). Each sampling method has its advantages depending on the specific project objectives. Criteria for evaluating common probability sampling designs in the context of accuracy assessment are discussed in detail by Stehman (1999) and other sampling issues are reviewed in Foody (2002).

Clearly, the choice of a sampling method for training data selection will impact the sample's ability to capture the spectral variability within the image. Unfortunately, while sampling methods have been studied exhaustively with respect to accuracy assessment, which sampling method is best suited for the selection of training data remains an open question. There is also a need to determine how sampling method and sample size interact and whether one method may be able to create accurate image classifications with fewer samples.

1.3 Training Data Editing

A number of methods have been proposed to improve the quality of training data sets (Sanchez et al. 2003) as described below. Arai (1992) used both clustering of training areas and edge image thresholding to create masks that would remove pixels in small areas that are spectrally separable from other pixels in the training area and exhibit high local spectral

variability. When applied to Landsat TM data for classification of an agricultural area with the maximum likelihood decision rule, overall accuracy improvements of 11.9 percent resulted.

Training sample units that are incorrectly labeled have a negative impact on classification results, especially with nearest neighbor methods (Cortijo and Perez de la Blanca 1997, Sanchez et al. 2003). Methods for editing training data sets that are used with the nearest neighbor decision rule have been explored by Cortijo and Perez de la Blanca (1997), Barandela and Gasca (2000), Barandela and Juarez (2002), and Sanchez et al. (2003). Not only do the editing procedures attempt to identify and remove problem sample units, but several methods, including generalized edition and depuration, allow a sample unit to be fixed, i.e. relabeled, instead of eliminated. Multiple iterations of a single or combination of editing method(s) are used to obtain edited training data samples that often result in classifications of higher accuracies (Sanchez et al. 2003). Although correction of errors or noise in the data sets is a primary goal of many editing procedures, similar methods are also used to weed training data samples of sampling units that do not add to the discriminatory ability of the classifier (Dasarathy et al. 2000, Raicharoen and Lursinsap 2005).

1.4 Classification Decision Rule

The particular decision rule used to assign pixels an informational class influences which sample characteristics are most important. With parametric classification techniques, spectral classes are assumed to follow multivariate normal distributions, and typically require estimation of both a mean vector and covariance matrix for each class. Nonparametric classification methods, including nearest neighbor, neural networks, and decision trees, can be accurate alternatives to traditional, parametric methods, like maximum likelihood (Cortijo and Perez De

La Blanca 1997, Friedl and Brodley 1997). These methods obviate the need for normally distributed class statistics.

Training data sets, having the same proportion of samples in each informational class as the image being classified, resulted in more accurate classifications with nearest neighbor techniques than with parametric methods (Hardin 1994). Also, numerous applications of the nearest neighbor decision rule in the forestry sector have shown the utility of this method (Franco-Lopez et al. 2001, Katila and Tomppo 2001, Makela and Pekkarinen 2001, McRoberts et al. 2002b, Nilsson et al. 2003, Tomppo and Halme 2004).

1.5 Band Combinations

The process of selecting a subset of bands for use in a classification procedure is referred to as feature extraction or selection (Campbell 2002). Feature selection can occur on the raw bands alone, or on any of a wide variety of derived bands, including principal components, vegetation indices, or band ratios. Reduction of data dimensionality is one of the major goals of feature selection. Feature selection is used mainly to improve the ability of a classifier to accurately discriminate between informational classes. A subset of bands is selected in which the classes of interest are spectrally most separable. Several methods have been suggested in the literature for feature selection, including spectral separability indices and band selection methods. The aforementioned feature selection techniques, information content measures, and classification accuracy predictors are explored in more detail below.

1.5.1 Spectral Separability Measures

When using parametric statistical methods for the assignment of informational classes to the pixel brightness value vectors within an image, the two most widely used and recommended measures of spectral class separability are transformed divergence (TD) and Jeffries-Matusita

(JM) distance (Mausel et al. 1990, Jensen 2005). Since these separability statistics require both a mean vector and covariance matrix for each spectral class within the informational classes of interest, they are not compatible with the minimum distance to means parametric classifier or nonparametric classifiers, such as nearest neighbor. Euclidian distance is the most commonly used separability metric for these other procedures (Richards and Jia 1999).

1.5.2 Band Selection Techniques

Viable alternatives to separability measures are the traditional backward, forward, and stepwise selection methods used for variable selection with multiple linear regression (Mather 2004). Aha and Bankert (1996) found feature selection methods to work better than a separability index when used with a nearest neighbor classifier. An issue that has been raised with both separability measures and selection methods is that they are not guaranteed to select the optimum combination of bands (Yool et al. 1986, Mather 2004).

1.5.3 Information Content Measures

Several techniques, including the optimal index factor and the Sheffield index, have been introduced to look at the information content of an image instead of only considering the spectral separability between informational classes in the training data (Chavez et al. 1982, Sheffield 1985, Beauchemin and Fung 2001). In a comparison of these two indices, Beauchemin and Fung (2001) found very different results when applying each to the same dataset. Hence neither index is a replacement for techniques that optimize the band combination for a given application. Although these methods are potentially useful for the purpose of data reduction, they were designed more for the selection of band combinations that result in better image interpretations than for optimizing classification procedures.

Although principal components analysis (PCA) is a widely applied method for reducing data dimensionality while retaining the maximum information content (Campbell 2002), the resulting bands are not a function of inter-class differences (Mather 2004). Thus the use of PCA may not result in improved classification accuracy. Canonical correlation analysis can provide more appropriate band transformations for maximizing the separability between classes (Lillesand et al. 2004).

1.5.4 Classification Accuracy Prediction

Although both separability indices and band selection techniques attempt to select a band combination resulting in improved classification accuracy, they do not provide an estimate of the classification accuracy. A common method for estimating classification accuracy is known as the leave-one-out or cross-validation method. This technique uses the training data sample to estimate the classification accuracy by leaving out a single sampling unit and classifying that unit with the rest of the sample. Repeating this deletion for every unit in the sample allows the percentage of correct classifications to be calculated. Despite the removal of each unit from the sample prior to classification, this method usually results in an optimistic estimate of accuracy or the misclassification error rate (Nadler and Smith 1993).

1.6 Summary

Based on the literature:

- Single pixel training is preferred over contiguous pixel training from the standpoint of statistical independence.
- Larger training data sample sizes tend to result in more accurate classifications, but the relationship may be influenced by sampling method, image and class heterogeneity, and classification algorithm.
- Smaller training data samples may yield classifications as accurate as those produced using larger samples.
- Frequent violation of the assumption of multivariate normality in the distribution of spectral classes limits the usefulness of parametric classifiers for many applications.
- A robust comparison of sampling methods for training data collection has not been reported in the literature.
- The interaction between training data sample size and sampling method warrants further exploration.
- All available image bands are not always needed and, in some cases, may result in lower classification accuracies.
- Spectral Euclidian distance is the most appropriate separability index for use with nearest neighbor classifiers.
- Use of separability indices or band selection techniques does not guarantee the choice of a band combination that will maximize classification accuracy.
- Cross-validation accuracies have potential as predictors of classification accuracies.

- The appropriate number of bands for use in a classification is influenced by the size of the training data sample.
- Training data editing can produce improvements in classification accuracies.

Given the constraints of previous efforts, in terms of a minimal number of realizations and relatively cursory examinations of the impact of confounding factors on the resulting classification accuracies, there is a need for a robust simulation-based approach to explore the interaction of training data sample size and sampling method on the accuracy of medium resolution image classifications and the resulting forest area precision estimates. Although previous studies have looked at the best band combination for a given application, these studies usually were based on a single training data sample. Since classification accuracies are influenced by the choice of band combination, the interaction between training sample properties and appropriate band combinations also require further exploration.

1.7 Objectives

The objectives of this study were as follows:

1) To determine the impacts of sampling method and sample size on the classification accuracy of forest/nonforest Landsat ETM+ image classifications using single pixel training with a nearest neighbor decision rule.
2) To determine the variability in forest area estimates and forest area precision estimates from a distribution of independent classifications.
3) To determine the impact that editing the training data pool to remove pixels that are spectrally unrepresentative of their informational class has on classification accuracies.
4) To determine if all six non-thermal Landsat ETM+ bands are needed to create accurate forest/nonforest classifications.
    a. If all six bands are not necessary, determine an appropriate band subset size.
    b. Determine which band combinations, of a given number of bands, are most useful in differentiating between forest and nonforest.
5) Explore the utility of separability indices and cross-validation techniques for the prediction of classification accuracy.

Table 1. Comparison of Studies that Explored Training Data Sample Size

Study	Imagery	Classification Scheme	Classification Method(s)	Training Sample Unit	Training Data Sample Size(s)	Training Sampling Method	Replicates per Sample Size
Maselli et al. 1992	Landsat TM; 30m spatial; 6 spectral bands	coniferous forest, deciduous wood, cultivation of cereals, olive grove, and urban area	maximum likelihood (ML), nonparametric (NP), and ML with priors from (NP)	single pixel	250, 500, 1000, 2000, and 4000 total	stratified random	10
Dobbertin and Biging 1996	3 simulated images; 25m spatial; 6 spectral bands	16 forest types (4 crown cover classes x 4 tree crown sizes)	maximum likelihood	single pixel and square of contiguous pixels	9, 16, 25, 49, and 100 per class	stratified random and stratified cluster	4
Arora and Foody 1997	Airborne Thematic Mapper; 11 spectral bands	sugar beat, wheat, barley, carrots, potatoes, and grass	discriminant analysis, fuzzy *c*-means, and neural network	single pixel	10, 30, 50, and 100 per class	stratified random	1
Chen and Stow 2002	6 Digital Orthophoto Quarter Quadrangles; 2-16m spatial; 3 spectral bands	single-family, multi-family, industry, irrigated grassland, cleared land, and undeveloped land	maximum likelihood	single pixel, contiguous pixels (seed), and polygons/blocks	25, 50, and 100, or 15 and 25 seeds, or 15 and 25 small blocks per class	none, user specified locations	1
Pal and Mather 2003	Landsat ETM+ (30m/6 bands) and hyperspectral DAIS (5m/ 72 bands)	wheat, potato, sugar beet, onion, peas, lettuce, and beans	maximum likelihood, neural network, univariate decision tree, and multivariate decision tree	single pixel	100, 150, 200, 250, 300, 350, and 400 per class	random	1
Foody and Mathur 2004a	Airborne Thematic Mapper; 5m spatial; 11 spectral bands	sugar beat, wheat, barley, carrot, potato, and grass	SVM, discriminant analysis, decision tree, and neural network	single pixel	15, 30, 45, 60, 75, 90, and 100 per class	stratified random	5
Foody and Mathur 2004b	SPOT	winter wheat and spring barley	SVM	single pixel	75 per class	not specified	1

Chapter 2
METHODS

2.1 Study Area

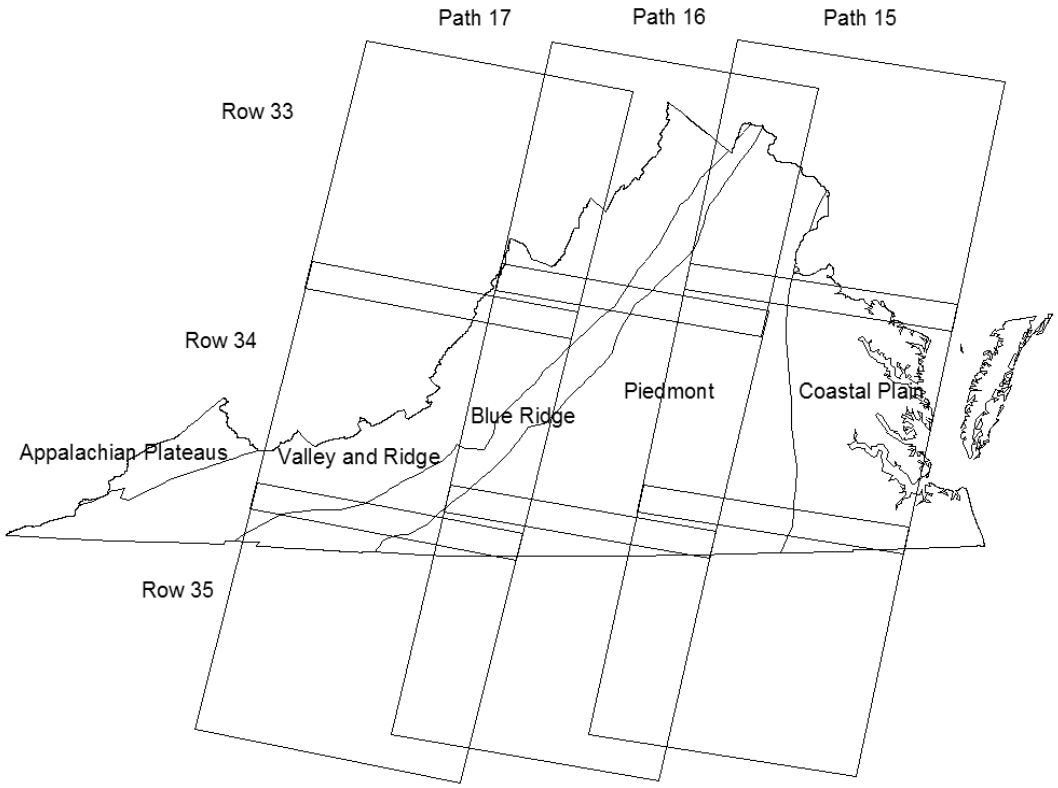
The study area covered all but the southwestern portion of the Commonwealth of Virginia, USA, with the center of the study area at approximately 37° 28' N latitude and 77° 20' W longitude. Virginia has a temperate climate and a diversity of physiographic provinces. The physiographic provinces within the study area included from east to west: the Coastal Plain; Piedmont; Blue Ridge; and Ridge and Valley (Figure 2). The following descriptions are based on information from Fleming et al. (2004).

The Coastal Plain consists of very little topographic variation with a maximum elevation around 75 meters, which slopes slightly downward toward the Atlantic Ocean. Forest vegetation within the province consists of managed loblolly pine stands and other pine-hardwood forests including longleaf pine-scrub oak with a few remnant beech-oak stands. Extensive wetlands, including four large river systems, the Potomac, Rappahannock, York, and James, are present within the Coastal Plain. The influences of the Atlantic Ocean and river systems result in the eastern part of the state having a warmer climate and longer growing season than the western areas.

A more rolling or hilly topography is characteristic of the Piedmont with elevation variations between 50 and 300 meters. The area has been described as a patchwork of secondary forest, pastures, and fields resulting from a history of clearing, agriculture, logging, and anthropogenic disturbances (Fleming et al. 2004). Forest vegetation includes Virginia pine, tulip-poplar, shortleaf pine, loblolly pine, sweetgum, and oak-hickory.

Both the Blue Ridge and Ridge and Valley provinces are part of the Appalachian mountain system. A much higher degree of topographic variation exists in these

Figure 2. Physiographic Provinces within Virginia Overlaid with Landsat Scenes



provinces with the highest elevation of 1746 meters at Mount Rogers in the southern Blue Ridge, other ridges in the 1200-1400 meter range, and valleys below 900 meters. Forest vegetation within these areas is predominated by deciduous forests mainly of mixed oak and oak-hickory. Climatic variations due to topography have made it possible for northern hardwood species and red spruce to occur at very high elevations. The wide range of topography within Virginia has also resulted in a diverse climate with precipitation varying from as little as 84 cm in some of the valleys to greater than 152 cm in parts of the southwestern mountains. Average precipitation for the entire state is around 108 cm.

A majority of the large urban areas within Virginia, including the Northern Virginia Washington-DC area, Norfolk/Virginia Beach area, and Richmond, fall within the Piedmont and Coastal Plain. Population densities within the state are greatest in the north eastern and eastern

parts of the state with a general decrease in density with east to west and north to south movement. A decrease in the degree of fragmentation occurs as the land cover and land use change from a mixture of urban areas, agriculture, and forest in the east to more contiguous blocks of agriculture and forest land with smaller urban areas in the west. Griffith et al. (2003) reported an increase in forest fragmentation from 1972 to 2000 across the southeastern United States. The study area in the above work, which included three of the physiographic provinces within Virginia, found higher rates of urbanization in the Piedmont than in the Blue Ridge. It also described both the Coastal Plain and Piedmont as being more dynamic in terms of land use change.

2.2 Data

2.2.1 Imagery

Three mosaicked Landsat ETM+ satellite images covering all portions of the Landsat Worldwide Reference System (WRS) paths 15, 16 and 17 within Virginia were used in this study and are shown over the physiographic provinces in Figure 2. These images, which will be referred to as VA15, VA16 and VA17, were leaf-off and acquired on November 6, 2001, November 13, 2001 and March 28, 2002, respectively. Each scene had been mosaicked and reprojected to the Virginia Lambert Conformal Conic projection by the Virginia Department of Forestry using Virginia Base Mapping Program (VBMP), (Commonwealth of Virginia 2002), digital orthophotography, for reference coordinates of ground control points. A total of 52, 103, and 36 ground control points were selected, respectively, for VA15, VA16, and VA17, and used to perform rectifications with a second order polynomial, which resulted in root mean square errors (RMSE) of 8.929, 4.791, and 12.004 meters, respectively. Additional preprocessing of the imagery was limited to the manual removal of clouds and cloud shadows.

2.2.2 Training Data

Training data sample pools were created by the interpretation of FIA Phase 1 point locations from a systematic, hexagonal grid over Virginia Base Mapping Program (VBMP) true color, leaf-off, digital orthophotography, which were originally collected at a scale of 1:4,800 and later resampled to a 1 meter spatial resolution (Figure 3). The VBMP imagery was collected in the Spring of 2002, which is close to the dates of the Landsat imagery. The classification scheme used for the purpose of image interpretation consisted of four classes: forest; nonforest; census water; and noncensus water. FIA definitions of each land use (Table 2) were used by the Virginia Department of Forestry personnel that performed the image interpretations (USDA-FS 1998).

In addition to the minimum size requirements for each class, the forest class also had a stocking requirement of at least ten percent tree cover. Clearcuts are considered a forest land use if they are not being converted to another use. Since there was not a size requirement on urban land uses, points which fell on improved roads were labeled as nonforest even if the surrounding land cover was all forest. The process of interpreting the Phase 1 points over the VBMP imagery was carried out in ESRI, ArcMap 8.3, which allowed for the addition of buffers around each point to aid the interpreter in estimating the area of a feature (Figure 3) and provided a measurement tool for verification of widths.

2.2.3 Validation Data

Point locations at the center of FIA ground plots (USDA-FS 2005) were used to validate all classifications. The plot centers were located with coordinates from survey grade Geographic Positioning Systems (GPS) and had ground verified land use calls that corresponded with the FIA definitions in Table 2. Both the center land use call and the total proportion forest for each

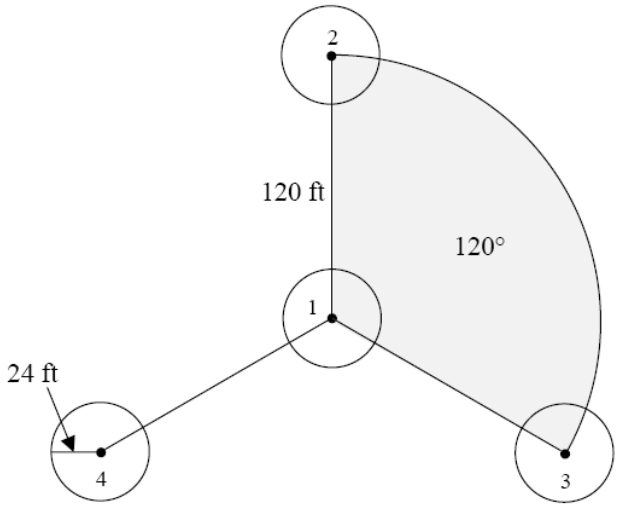
Table 2. Photo-interpreted Class Definitions and Minimum Mapping Units

Land Use	Minimum Width (meters)	Minimum Area (hectares)	Includes
Forest	36.576	0.4047	Clearcuts
Nonforest	36.576	0.4047	Agriculture, Marsh
Nonforest (Human-caused)	0	0	Improved Roads, Powerlines, Cultural Developments
Census Water	60.96	1.82	
Noncensus Water	9.144	0.4047	

Figure 3. Example of Phase 1 Point over VBMP Image with 35.89m Buffer

plot, which consists of four subplots (Figure 4), were used in accuracy assessment and forest area estimation procedures.

Figure 4. FIA Ground Plot Design with Four Subplots



2.2.4 Data Summary

Although both the training and validation points had more than two possible land use designations, they were grouped into forest and nonforest for all analyses. Presented in Tables 3 and 4 are: (1) the number of total and nonzero pixels per image, (2) the ground area covered by each image, (3) the sample size of each training data pool and validation dataset, (4) the proportion of forest points in each dataset, and (5) the average proportion of forest observed at each FIA ground plot.

Table 3. Image Size and Area Covered

Image	Total Pixels	Nonzero Pixels	Hectares
VA15	80,126,720	46,402,142	4,176,193
VA16	98,427,329	57,537,939	5,178,415
VA17	92,313,432	29,631,683	2,666,851

Table 4. Sample Size and Forest Proportions of Training Data Pools and Validation Sets

Image	Training Data (Pixels)	Proportion Forest Training	Validation Data (Pixels)	Proportion Forest Validation	Average Plot Proportion
VA15	46,903	0.5831	1,742	0.6332	0.6281
VA16	58,176	0.6222	2,152	0.6236	0.6216
VA17	29,937	0.6921	1,110	0.6703	0.6612

2.3 Simulation Steps

Figure 5 contains a flow chart of classification and validation stages 1 and 2.

2.3.1 Training Samples

Training data samples of sizes 25, 50, 75, 100, 200, 300, 400 and 500 were drawn from the training data pool of each image using either random, stratified random or systematic sampling. Sampling with replacement was used for both the random and stratified random sampling methods. The stratified random samples were stratified by informational class (i.e., forest or nonforest) with proportional allocation. The proportion of forest and nonforest pixels in each image's training data pool was used to calculate the appropriate proportion for allocation of the stratified random samples. Since the training data were interpreted from a systematic hexagonal grid across the entire study area and the exact ground proportion of forest was not known, the proportion of forest in the training data pool was deemed a reasonable estimate of the ground proportion. Approximately one thousand training data samples for each image, sample size, and sampling method combination were created.

Since one thousand unique samples were not possible with larger sample sizes and the systematic sampling method, all possible samples were created and evaluated for this method. Although this sampling method is referred to as systematic, it did not create samples with

identical spatial distances between adjacent pairs of training pixels. The training points for the systematic sampling runs were sorted first by the row number and then by the column number of the corresponding image pixel (Table 5). For each sample size, the training data pool size was divided by the sample size and the quotient rounded down to obtain a skip number. The first systematic training sample draw started with the first training data pixel, and then sampled every n^{th} pixel, with n equal to the skip number, until the desired sample size was achieved. Subsequent draws started with the training data point corresponding to the draw number until the maximum number of draws were created (i.e., the last sample included the last training data point). The main goal of using systematic sampling was to assure relatively even coverage of all image areas, which this pseudo-systematic method achieved.

Table 5. Example of Training Data Input Information for Systematic Sampling

Image Row	Image Column	Location	X	Y	Class
43	2919	320103	156679	370040	Forest
52	2852	388004	154654	369767	Forest
56	2818	418178	153643	369632	Forest
65	2974	486302	158324	369379	Nonforest
69	2940	516476	157310	369246	Nonforest

2.3.2 Classification Stage 1

In the first classification stage, each training data sample was used to classify the validation set of the corresponding image using all six non-thermal ETM+ bands. Each pixel within a training sample was treated as a separate spectral class and used in a nearest neighbor classification with one neighbor. The sum of the squared differences (SSD) between each band in the brightness value vector of the pixel being classified and the brightness value vector of the

spectral class under consideration (Eq. 1) was used. SSD was used in lieu of Euclidian distance to minimize the number of calculations. It was calculated as follows:

$$SSD = (\mathbf{x}-\mathbf{m})'(\mathbf{x}-\mathbf{m}) \quad (1)$$

where $\boldsymbol{x}$ is the brightness value vector of the pixel being classified, $\boldsymbol{m}$ is the brightness value vector of the spectral class under consideration and $\boldsymbol{(x-m)'}$ is the transpose of the difference vector, was used to assign the appropriate informational class to each pixel being classified.

2.3.3 Validation Stage 1

Accuracy assessments were performed on the first classification outputs, i.e., the classified validation data. The following were calculated and/or recorded for each classification:

- error matrix
- overall classification accuracy
- forest sample size in training sample
- nonforest sample size in training sample.

In addition, the following were calculated and/or recorded for each validation point per classification:

- location
- informational class assignment
- spectral class assignment
- spectral distance (Eq. 1) to its spectral class.

2.3.4 Classification and Validation Stage 2

Although classification of the validation data for each image with a large number of training samples was a computationally efficient way to explore trends in classification accuracies, it did not allow exploration of the trends in forest area estimates, which require full

image classifications. Since creating a full image classification for every training data sample draw for each of the 72 combinations of image, sampling method, and sample size was computationally too expensive, a subset of 101 sample draws from each of the 72 combinations were selected to create full image classifications. Selection of these training samples was accomplished by sorting the overall classification accuracy (based on the validation data set) of each sample draw from low to high and then selecting the sample draws with the lowest and highest accuracy and every n^{th} draw in between these two extremes in order to cover the full range of accuracies. The n was 10 for both random and stratified random sampling, but varied based on the number of sample draws for systematic sampling.

Accuracy assessments of the whole image classifications were performed using the validation data set and for a given sample draw are identical to the accuracy assessment of the classified validation set in stage 1.

2.3.5 Forest Area Estimation

Forest map marginals, forest area estimates, and forest area precision estimates for all full image classifications were calculated using both the Card (1982) method of adjusting the map marginals with plot center land use and Cochran's (1977) stratified estimation with plot proportion forest. Forest map marginals are simply the proportion of the satellite image classified as forest, which is calculated by dividing the total number of non-zero pixels in an image that were classified as forest by the total number of non-zero pixels in the image. With the Card (1982) method, the forest map marginal from a classification is corrected based on the error matrix obtained from the validation data, which in this application were FIA ground plots. Given an error matrix of the form found in Table 6, the Card adjusted proportion forest (P_f) would be calculated as follows:

$$P_f = \left[M_2 * \left(\frac{n_{22}}{(n_{12} + n_{22})} \right) \right] + \left[M_1 * \left(\frac{n_{21}}{(n_{21} + n_{11})} \right) \right] \quad (2)$$

where M_i is the map marginal for class i (i = 1 for nonforest and i = 2 for forest) and n_{jk} are the error matrix elements from Table 6 with j referring to the reference data and k referring to the classified data.

Table 6. Example Error Matrix

Reference Data	Classified Data	
	Nonforest	Forest
Nonforest	n_{11}	n_{12}
Forest	n_{21}	n_{22}

The variance of the Card adjusted proportion forest was calculated using equation 3:

$$Var(P_f) = \frac{\left[M_2 - \left(\frac{M_2 * n_{22}}{(n_{12} + n_{22})} \right) \right] * \left(\frac{M_2 * n_{22}}{(n_{12} + n_{22})} \right)}{(M_2 * N)} + \frac{\left[M_1 - \left(\frac{M_1 * n_{21}}{(n_{11} + n_{21})} \right) \right] * \left(\frac{M_1 * n_{21}}{(n_{11} + n_{21})} \right)}{(M_1 * N)} \quad (3)$$

where N is the total number of validation points.

The equations based on Cochran's stratified estimation (1977) were obtained from McRoberts et al. (2002b) and are presented below (Eq. 4-6):

$$P_f = \sum_{j=1}^{J} W_j \hat{P}_j \quad (4)$$

$$Var(P_f) = \sum_{j=1}^{J} W_j^2 \hat{\sigma}_j^2 / n_j \quad (5)$$

$$\hat{\sigma}_j^2 = \frac{1}{n_j - 1} \sum_{i=1}^{n_j} (P_{ij} - \hat{P}_j)^2 \tag{6}$$

where P_f is the adjusted proportion forest; j is the stratum (in this case forest or nonforest); W_j is the weight of the jth stratum (in this case, the map marginals of each class); $\hat{P}_j$ is the average proportion of forest from all of the plots assigned to the jth stratum; n_j is the number of plots in stratum j; P_{ij} is the plot proportion forest of the ith plot in stratum j; and $\hat{\sigma}_j^2$ is the within-stratum variance of the jth stratum. McRoberts et al. (2002b) notes that equation 5 does not take into account the slight effects of finite population correction factors or the fact that the number of plots per stratum were variable instead of fixed.

The Forest Service scales the precision of forest area estimates based on a fixed reference area of 404,694 ha (1 million acres) in order to intercompare states and regions of varying sizes. Scaled precision for both methods was calculated using the following equation, which again was modified from McRoberts et al. (2002b):

$$\text{Scaled Precision} = \frac{\sqrt{Var(P_f)}}{P_f} \sqrt{\frac{A * P_f}{404694}} \tag{7}$$

where A is the total image area calculated in hectares.

The major difference between the two estimation methods was that plot center land use was used in the Card estimation procedure and plot proportion forest was used in the Cochran estimation procedure. Although it is possible to use plot center land use with the Cochran method and plot proportion forest with the Card method, the methods used in this study correspond with those currently used in practice.

2.3.6 Outputs from Whole Image Classifications

For each whole image classification, the following information was calculated and recorded:

- draw number from the full set of training samples
- error matrix
- overall classification accuracy
- forest map marginal.

Elements specific to the Card calculation were:

- forest area estimate
- variance of the forest area estimate
- precision of the forest area estimate.

Elements specific to the Cochran calculation were:

- mean plot proportion forest of the forest stratum
- mean plot proportion forest of the nonforest stratum
- forest area estimate
- total area in forest
- within-stratum variance of the forest stratum
- within-stratum variance of the nonforest stratum
- sum of the within-stratum variances
- variance of the total forest area
- precision of the forest area estimate
- total number of validation points classified as forest
- total number of validation points classified as nonforest.

2.3.7 Classification and Validation Stage 3: Band Combinations

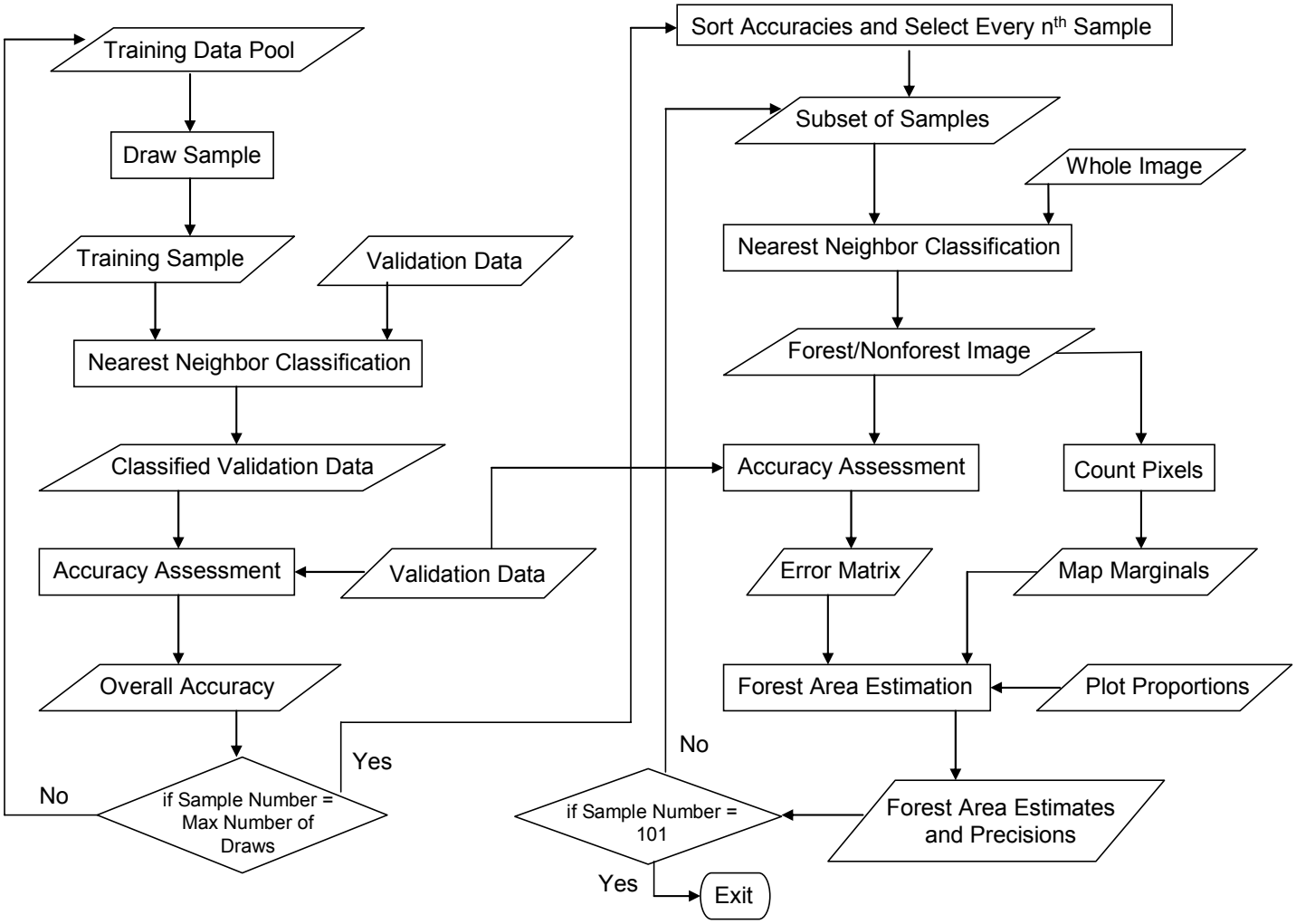
The validation data were classified as forest or nonforest using all sixty-three possible band combinations of the six, non-thermal Landsat ETM+ bands for each training data sample created in section 2.3.1. (Note: Landsat ETM+ band 6, the thermal infrared band, was not used in this study and mid-infrared band 7 is referred to as band 6 through out this work.) Classification accuracies were computed for each band combination and sorted to determine the band combination that produced the highest overall classification accuracy for each training data sample draw. The number of times a given band combination produced the most accurate classification was summarized across all samples for each image, sampling method, and sample size combination.

A leave-one-out cross-validation procedure, with all possible band combinations, was used to calculate the accuracy with which each pixel in a training data sample would be classified based on the other pixels in the sample. Additionally, both the minimum and average Euclidian distances between pixels from different informational classes within each training data sample were calculated for all sixty-three possible band combinations. These two spectral separability measures and the cross-validation accuracies were compared to the overall classification accuracies with the same band combination. The purpose of this comparison was to determine if any of these three measures could predict the classification accuracies.

2.3.8 Classification and Validation Stage 4: Training Data Pool

Each training data sample created in section 2.3.1 was used to classify the training data pool of the sample's corresponding image. Photo-interpreted land use calls (2.2.2) for each training data pixel were compared with the pixel's class assignment and used to calculate classification accuracies. These accuracies were adjusted for sample size, using equation 8, in

Figure 5. Flow Chart of Simulation Stages 1 and 2



order to eliminate bias when comparing different sample sizes as the training samples were subsets of the pool.

$$Accuracy_{adj} = \frac{(n_{11} + n_{22} - s)}{(n - s)} \quad (8)$$

where $Accuracy_{adj}$ is the adjusted accuracy; n_{jk} are the error matrix elements from Table 6 with j referring to the reference data (photo-interpretation) and k referring to the classified data; n is the total number of reference pixels (training data pool size); and s is the training data sample size.

In addition to the values that were calculated and/or recorded for each validation point in 2.3.3, two additional values were calculated and/or recorded when the photo-interpretations were used as reference data:

- unique identifier of the training point associated with the spectral class
- spatial distance between the two training points (i.e., the one classified and the one in the current training sample).

The number of times the classification of each training data pixel disagreed with the photo-interpretation was summarized across all classifications.

2.4 Training Data Editing

Several methods were explored for identifying training data points that were potentially mislabeled. First, a 3x3 window was used to calculate the average spectral distance between each pixel in the training data pools and its eight neighbors in order to identify training data points that were in heterogeneous areas. Pixels with large spectral distances were considered for removal. Second, the likelihood of misclassification for each training data point was calculated by dividing the number of times the point was misclassified by the total number of classifications (only classifications with a sample size of 500 were used in this calculation, due to the higher

classification accuracies with this sample size). Training data points from VA17 were assigned to ten groups based on their misclassification rates. A random sample of ten training data points from each group were visually examined over the VBMP data. Based on this examination, a misclassification rate of sixty percent or greater was used to weed problem points from the training data pools for all three images. Before and after editing statistics for each training data pool are shown in Table 7.

2.5 Simulations with Edited Training Data

All of the steps described in section 2.3 were repeated with the edited training data pool with two changes. First, the number of training data samples drawn in 2.3.1 was one hundred instead of one thousand for each image and sample size combination with the random and

Table 7. Training Data Statistics: Before and After Editing

Image	Edited Training Data	Unedited Training Data	Proportion of Pixels Removed	Proportion Forest Edited	Proportion Forest Unedited
VA15	41,702	46,903	0.1109	0.6268	0.5831
VA16	51,651	58,176	0.1122	0.6746	0.6222
VA17	27,593	29,937	0.0783	0.7311	0.6921

stratified random sampling methods. All possible samples were still created with the systematic sampling method. Second, only eleven full image classifications (2.3.4) were created for each combination of image, sampling method, and sample size.

2.6 Other Analyses

The forest map marginals and forest area estimates from each classification were compared with the average proportion of forest on the FIA ground plots, the proportion of ground plots that had a center land use of forest and the proportion of phase 1 points (training

data points) that were interpreted as forest in each image. These surrogates or estimates of the true proportion forest on the ground were used as an additional indicator of a classification's quality.

The relative efficiency (RE) of each classification was calculated for both forest area estimation techniques. Relative efficiency measures the improvement in variance resulting from the use of double sampling (DS) or stratified analyses (SA) with image classifications, versus estimation just from ground plot data based on simple random sampling (SRS). The variance of the mean ground plot proportion forest with SRS is divided by the variance of the mean plot proportion with either DS or SA to calculate RE (Cochran 1977).

2.7 Implementation

All of these calculations were computed by programs that were created in Fortran 90 for implementation on a SGI Altix 3300 shared memory supercluster (Appendices A-F). Summary statistics and graphics were calculated and created using Minitab 14.

Chapter 3
RESULTS

3.1 Training Sample Size and Sampling Method Effects on Classification Accuracy

As the sample size of training data sets increases from 25 to 500 pixels, both the average and the median overall classification accuracy increase at a decreasing rate for classifications created with all six bands with all three images (Figures 6a-6c). Simultaneously, the standard deviation of overall accuracy decreases with sample size increases across all three images and sampling methods (Figures 7a-7c). This decline in the variability of classification accuracy with increasing sample size is also illustrated by the boxplots in Figures 8a-8c.

Figure 6a. VA15 Average Classification Accuracy with All Six Bands versus Sample Size for Each Sampling Method

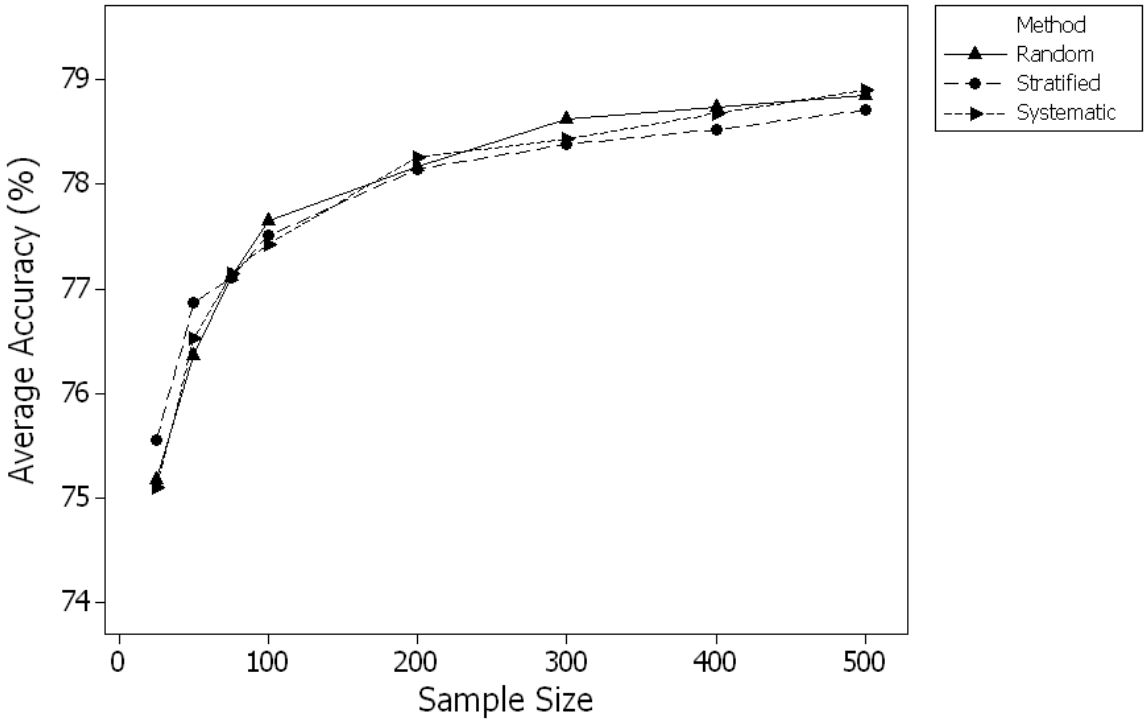


Figure 6b. VA16 Average Classification Accuracy with All Six Bands versus Sample Size for Each Sampling Method

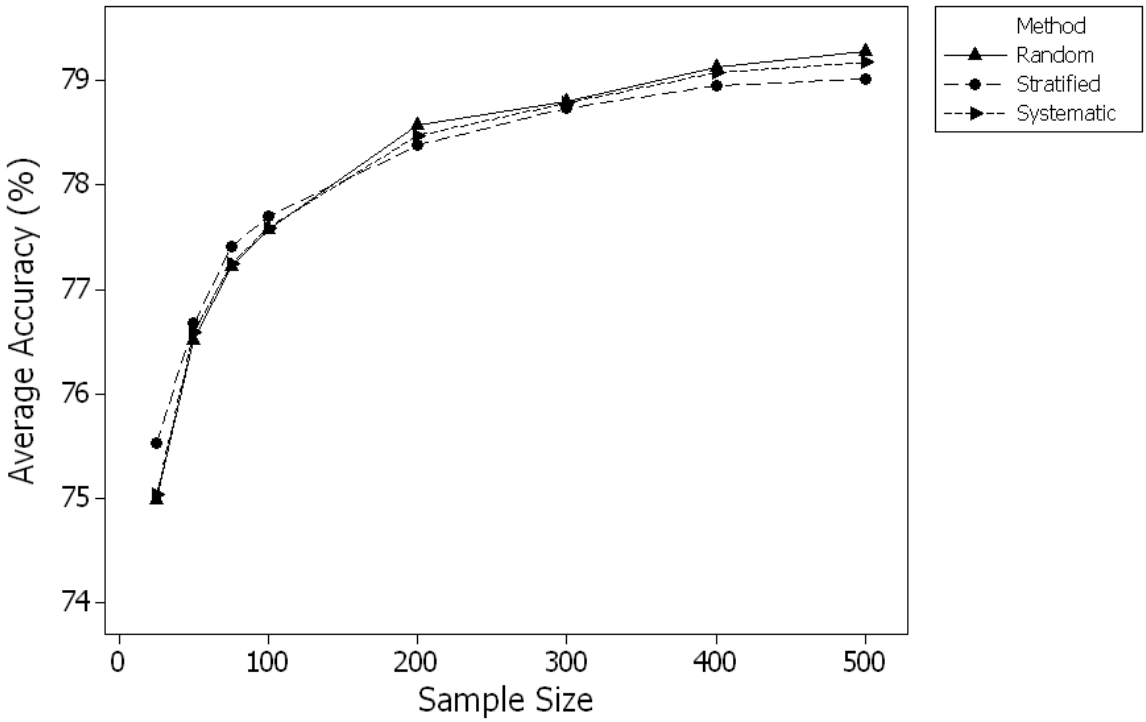


Figure 6c. VA17 Average Classification Accuracy with All Six Bands versus Sample Size for Each Sampling Method

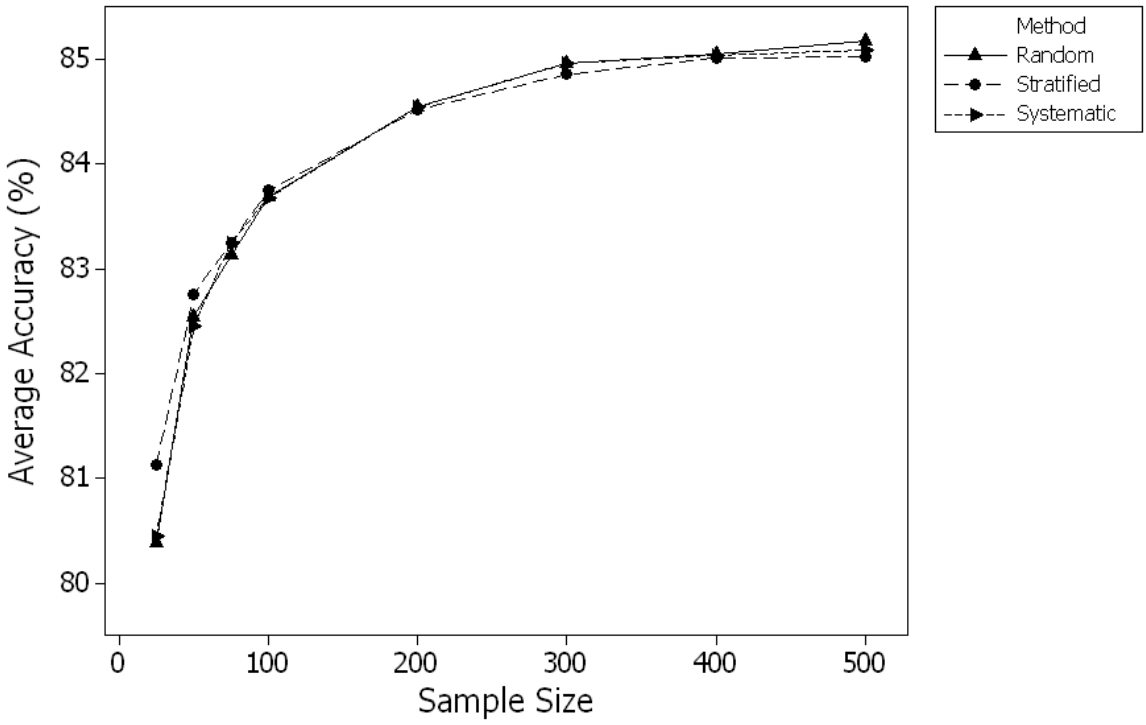
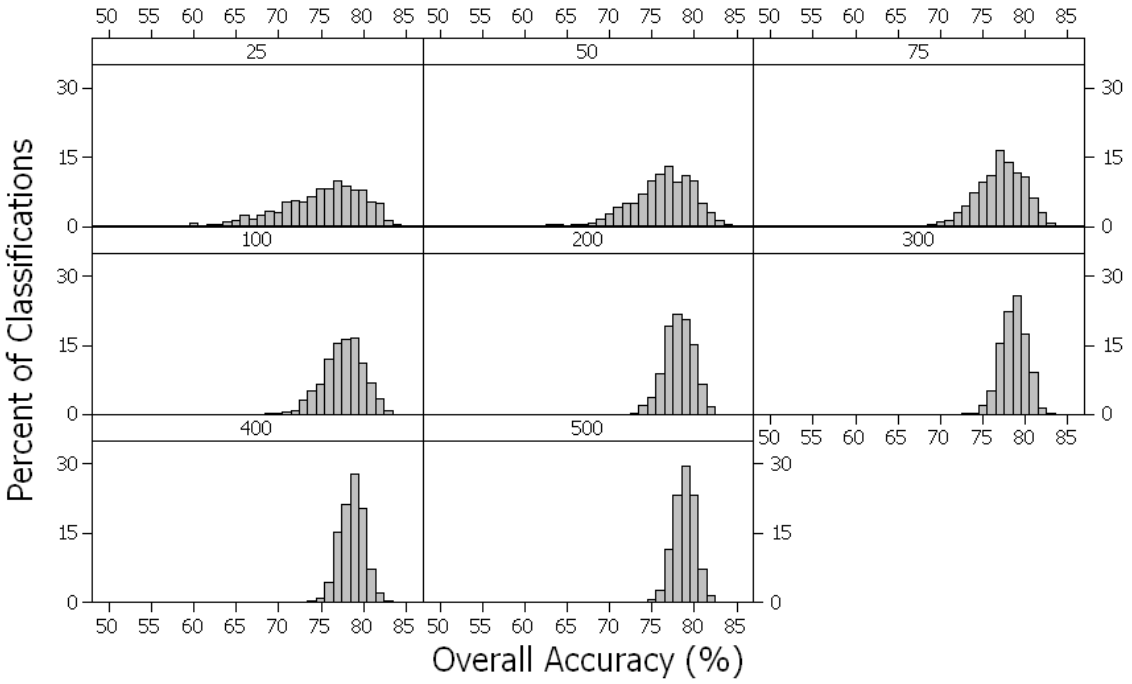
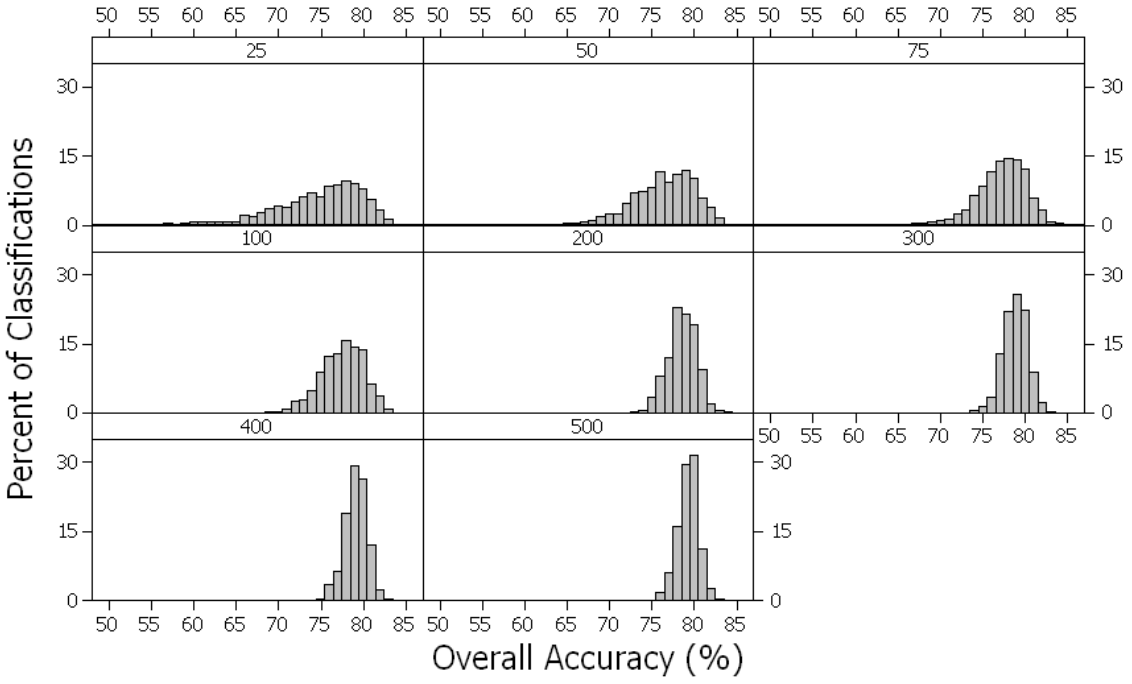


Figure 7a. Histogram of Classification Accuracies with All Six Bands by Sample Size for VA15 with Random Sampling



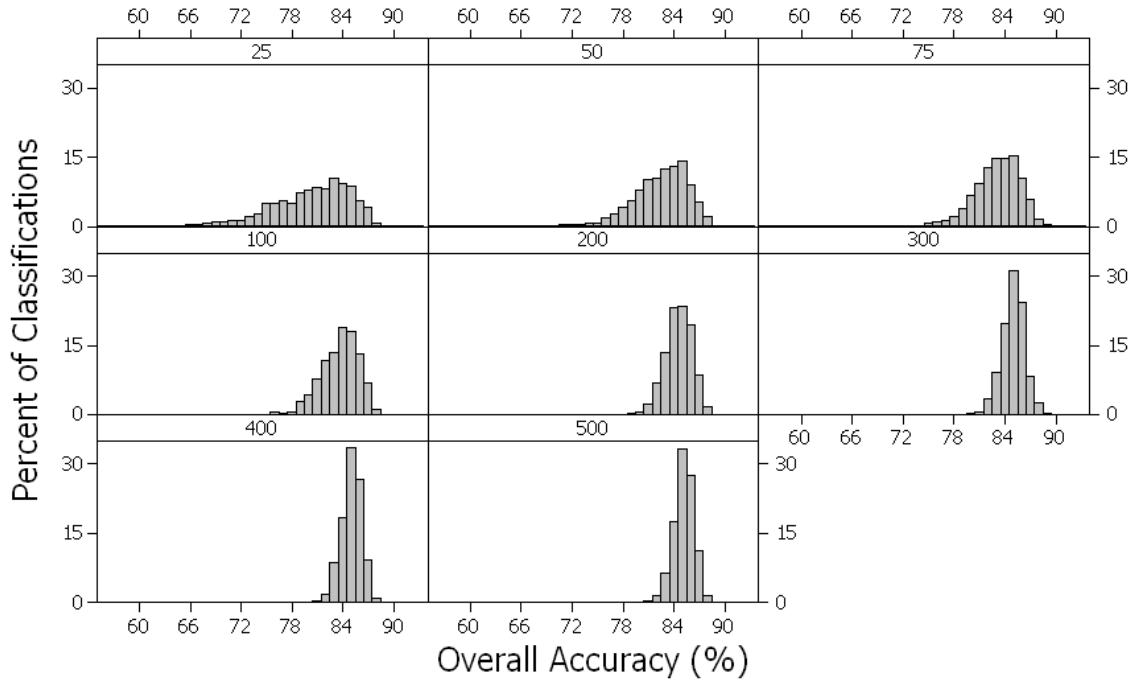
Panel variable: Sample Size

Figure 7b. Histogram of Classification Accuracies with All Six Bands by Sample Size for VA16 with Random Sampling



Panel variable: Sample Size

Figure 7c. Histogram of Classification Accuracies with All Six Bands by Sample Size for VA17 with Random Sampling



Panel variable: Sample Size

Figure 8a. Boxplot of Classification Accuracies with All Six Bands by Sampling Method and Sample Size for VA15

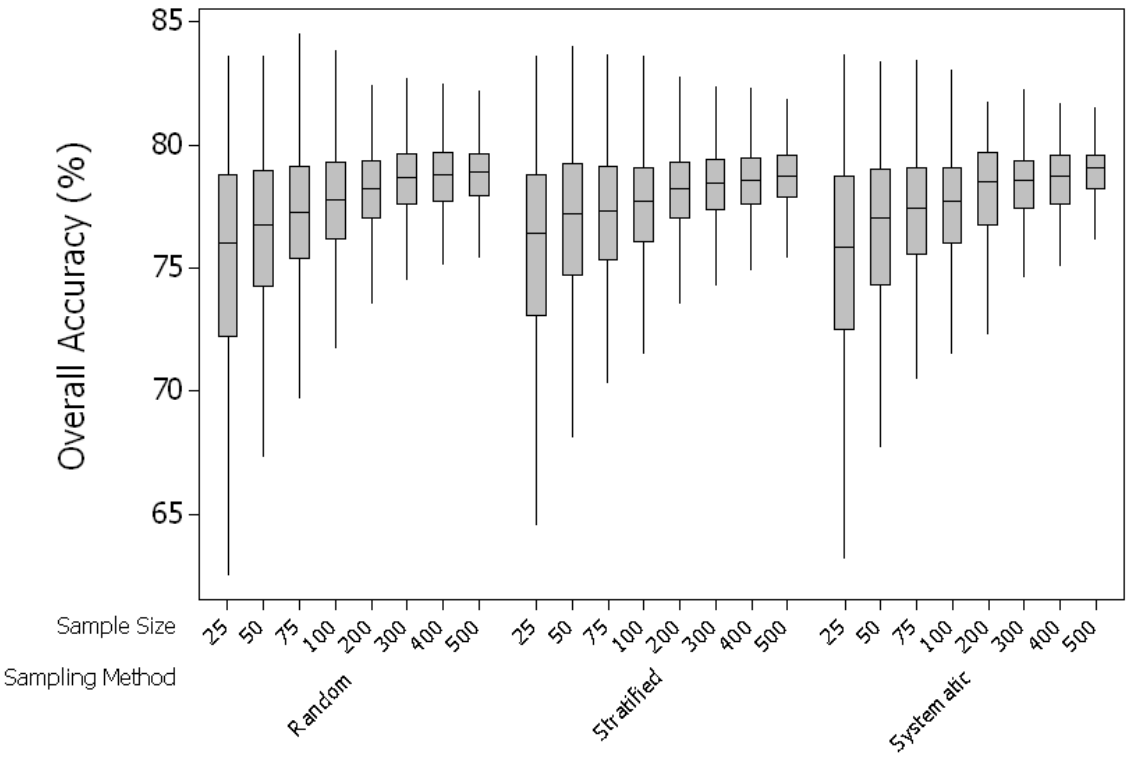


Figure 8b. Boxplot of Classification Accuracies with All Six Bands by Sampling Method and Sample Size for VA16

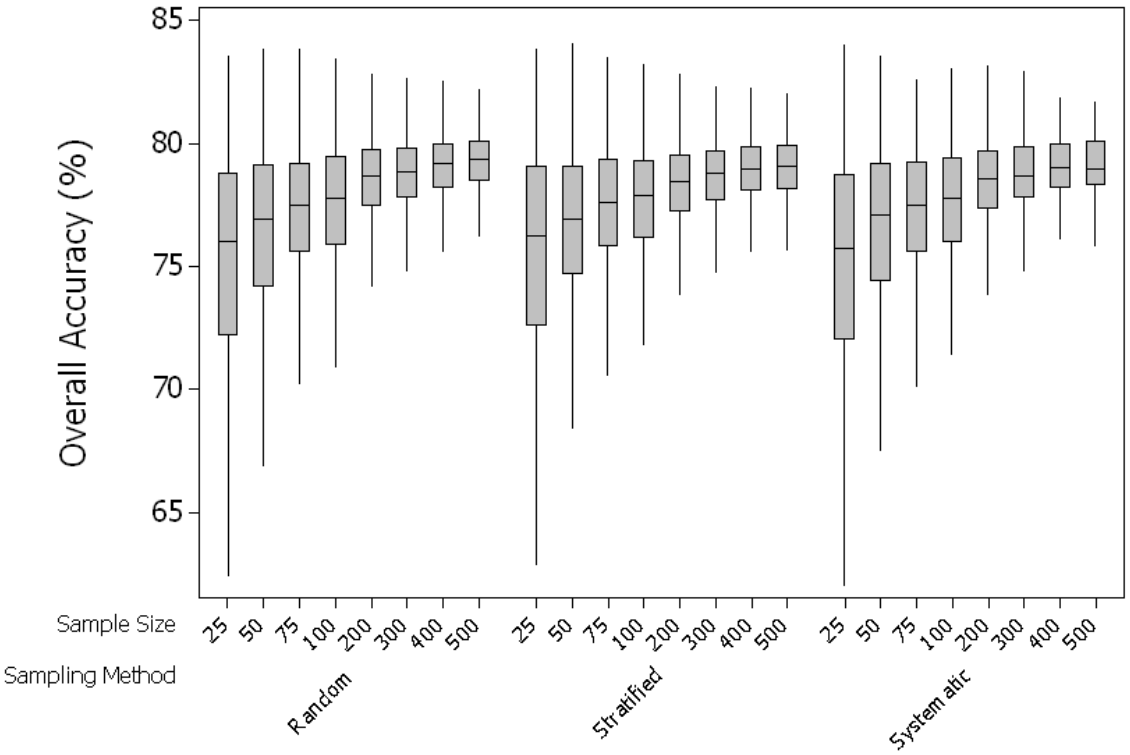
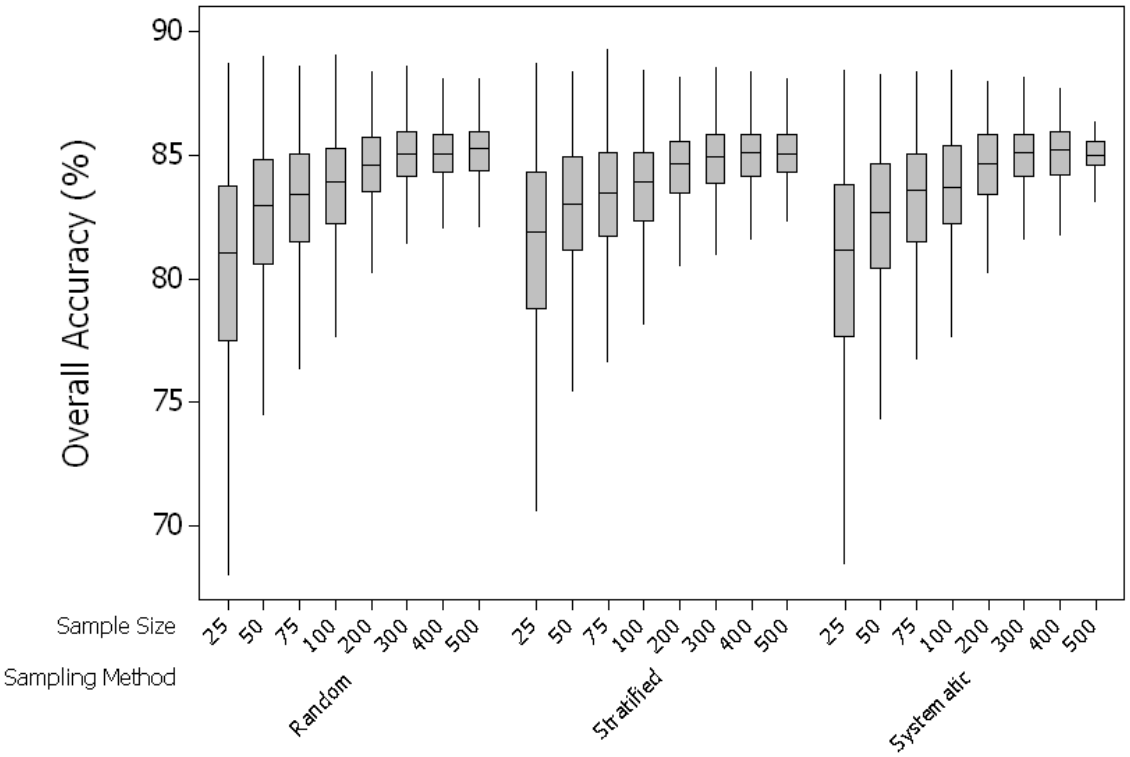
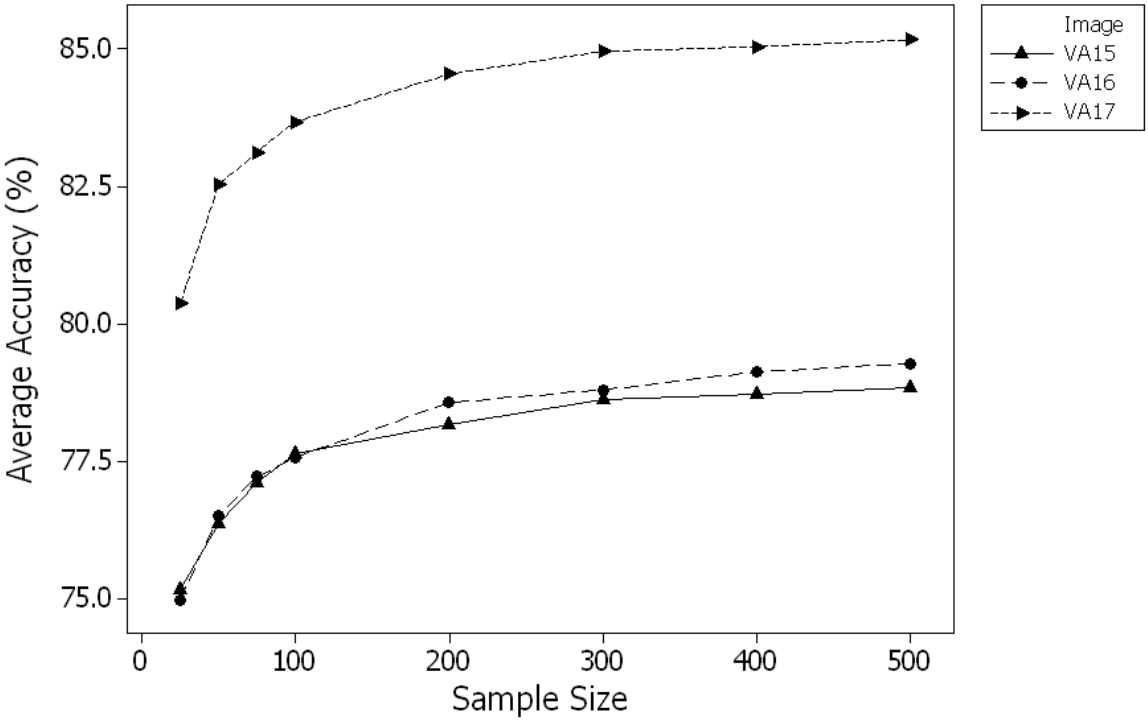


Figure 8c. Boxplot of Classification Accuracies with All Six Bands by Sampling Method and Sample Size for VA17



The sampling method which resulted in the highest average accuracy for a given sample size, varied with sample size (Figures 6a-6c). Although stratified random sampling resulted in the highest average accuracies when summarized across sample sizes for all three images, the maximum difference between the methods was only 0.104%. Samples created with the other two sampling methods only resulted in smaller average accuracies than stratified random samples when smaller sample sizes were used. However, noteworthy differences occurred in the average classification accuracies between the largest and smallest sample sizes, which were 3.57%, 4.02% and 4.46%, respectively for VA15, VA16, and VA17 (Figures 6a-6c). In addition, classifications of VA17 had significantly higher classification accuracies than both VA15 and VA16 (Figure 9).

Figure 9. Comparison of Average Classification Accuracy with Random Sampling versus Sample Size for All Three Images



Despite the increased likelihood of smaller sample sizes resulting in less accurate classifications, individual training data samples of sizes 50 and 75 resulted in the maximum overall accuracy of all 20,000 plus classifications per image. The user's and producer's accuracies for the classifications with the ten highest overall accuracies from each image are presented in Table 8. All three images had training data samples that produced high classification accuracies with small sample sizes.

3.2 Forest Area Estimation

As with classification accuracies, variability in the forest map marginals decreases as the training data sample size increases across all three images and sampling methods. In general, the average forest map marginal from classifications created with a given training data sample size tends to decrease and level out as the sample size increases (Figures 10a-10c). The forest map marginals of classifications created with stratified random samples were less variable than the map marginals from classifications created with the other two sampling methods (Figures 11a-11c). Despite the decreased variability of map marginals with stratified random sampling, forest area estimates had similar levels of variation across all three sampling methods and both estimation techniques, Card with plot center land use and Cochran with plot proportion forest (Figures 12a-12c).

Forest area estimates for a given image calculated with the Card method were always larger than the Cochran method (Figures 13a-13c). Precisions were also different between the two estimation methods. The average difference between the precisions with the Card and Cochran methods were 0.31% for VA15 and VA16, and 0.255% for VA17. Since VA17 had higher classification accuracies than VA15 and VA16, VA17 had more precise forest area estimates. The forest area precision estimates were directly related to classification accuracies,

Table 8. User's and Producer's Accuracies (%) for Classifications with the Ten Highest Overall Accuracies for Each Image

Image	Sampling Method	Sample Size	Overall Accuracy	User's Forest	User's Nonforest	Producer's Forest	Producer's Nonforest
VA15	Random	75	84.56	84.92	83.76	91.93	71.83
VA15	Stratified	100	84.27	85.34	82.07	90.75	73.08
VA15	Stratified	50	84.04	84.12	83.86	92.20	69.95
VA15	Random	100	83.87	84.77	81.96	90.84	71.83
VA15	Stratified	75	83.70	83.54	84.07	92.48	68.54
VA15	Systematic	25	83.70	82.58	86.60	94.11	65.73
VA15	Stratified	25	83.64	85.26	80.41	89.66	73.24
VA15	Stratified	100	83.64	84.43	81.95	90.93	71.05
VA15	Stratified	75	83.64	83.42	84.17	92.57	68.23
VA15	Random	25	83.64	82.72	85.98	93.74	66.20
VA16	Stratified	50	84.06	86.07	80.44	88.82	76.17
VA16	Systematic	25	84.01	83.31	85.63	93.00	69.14
VA16	Systematic	25	83.92	83.60	84.63	92.32	70.00
VA16	Random	75	83.88	84.57	82.47	90.69	72.59
VA16	Random	200	83.88	84.57	82.47	90.69	72.59
VA16	Random	50	83.88	84.43	82.74	90.91	72.22
VA16	Stratified	25	83.88	82.93	86.12	93.37	68.15
VA16	Random	200	83.83	86.07	79.85	88.38	76.30
VA16	Stratified	25	83.83	84.32	82.81	90.98	71.98
VA16	Systematic	25	83.78	84.94	81.53	89.94	73.58
VA17	Stratified	75	89.28	89.71	88.24	94.89	77.87
VA17	Systematic	300	89.19	88.90	89.94	95.83	75.68
VA17	Stratified	75	89.19	88.81	90.20	95.97	75.41
VA17	Random	100	89.10	89.38	88.40	95.03	77.05
VA17	Stratified	400	89.01	89.17	88.61	95.16	76.50
VA17	Random	50	89.01	89.07	88.85	95.30	76.23
VA17	Stratified	200	88.74	88.54	89.25	95.56	74.86
VA17	Stratified	75	88.74	88.35	89.77	95.83	74.32
VA17	Random	25	88.74	87.24	93.19	97.45	71.04
VA17	Stratified	25	88.74	86.98	94.14	97.85	70.22

with more precise estimates resulting from more accurate classifications. Thus, improvements in forest area precision estimates and a decrease in their variability was observed with increasing sample size, but no major differences occurred among sampling methods (Figures 14a-14c). Relative efficiencies are presented in Table 9, for both estimation methods with all whole image

classifications. Only classifications with extremely low accuracies resulted in variances of the mean proportion forest that were lower than the variances from just the FIA ground plot data.

Table 9. Statistics for Relative Efficiencies with Both Estimation Methods

Image	Estimation Method	Average	Minimum	Maximum	Number of Classifications
VA15	Card	1.180	0.863	1.534	2424
VA16	Card	1.216	0.870	1.533	2424
VA17	Card	1.483	0.897	2.046	2424
VA15	Cochran	1.396	0.995	1.898	2424
VA16	Cochran	1.436	0.999	1.836	2424
VA17	Cochran	1.759	1.007	2.510	2424

With VA15 and VA16, both estimation methods resulted in area estimates that on average were greater than the proportion of forest points in the training data pool. The Cochran estimates were closer than the Card estimates to this proportion. The opposite was true for VA17, where both estimation methods resulted in area estimates that on average were less than the proportion of forest points in the training data pool. With VA17, the Card area estimates were closer to this proportion. Forest map marginals from classifications of VA15 and VA17 with larger sample sizes, 200 or greater for VA15 and 400 or greater for VA17, were closer to the proportion of forest points in the training data than the forest area estimates from either estimation method.

3.3 Training Data Editing

Classifications with samples from the edited training data pools resulted in higher average classification accuracies across all seventy-two combinations of image, sampling method, and sample size than samples from the unedited pools. The minimum increase in

Figure 10a. VA15 Average Forest Map Marginal (Proportion) versus Sample Size for Each Sampling Method

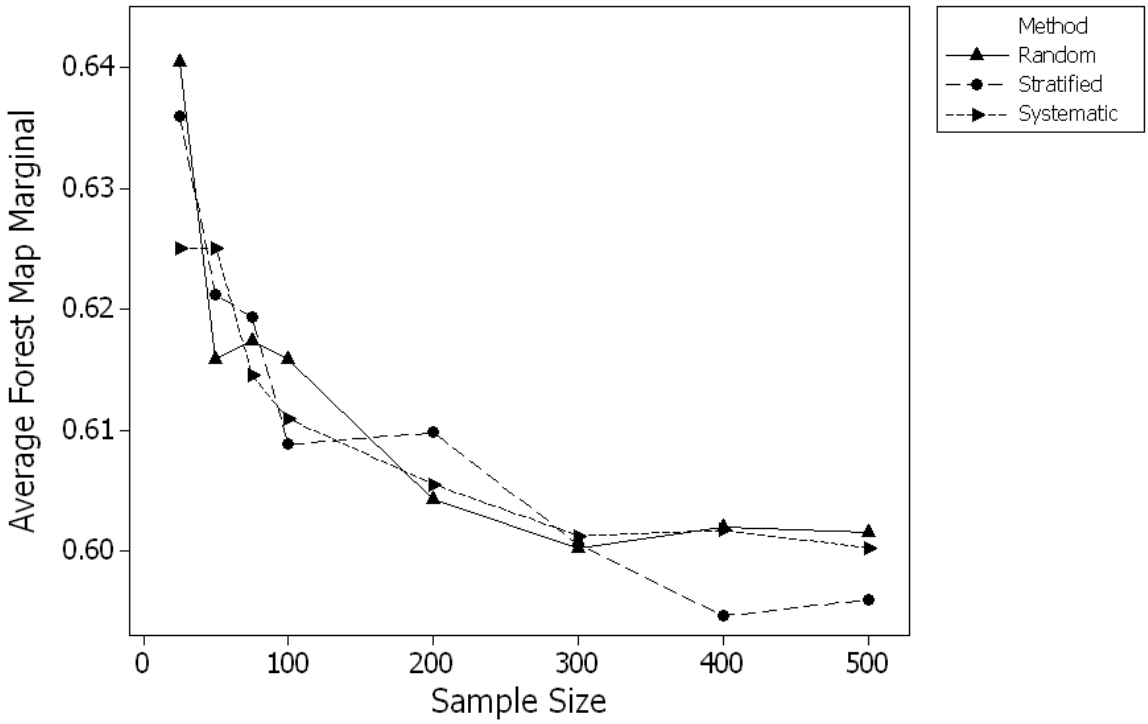


Figure 10b. VA16 Average Forest Map Marginal (Proportion) versus Sample Size for Each Sampling Method

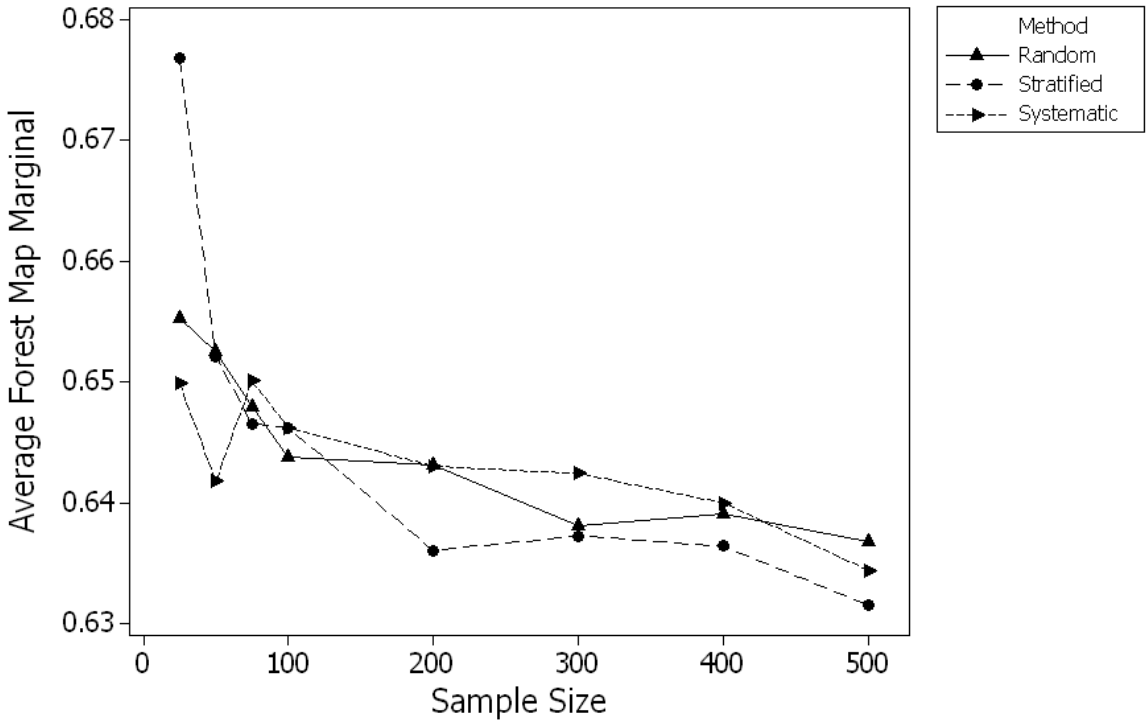


Figure 10c. VA17 Average Forest Map Marginal (Proportion) versus Sample Size for Each Sampling Method

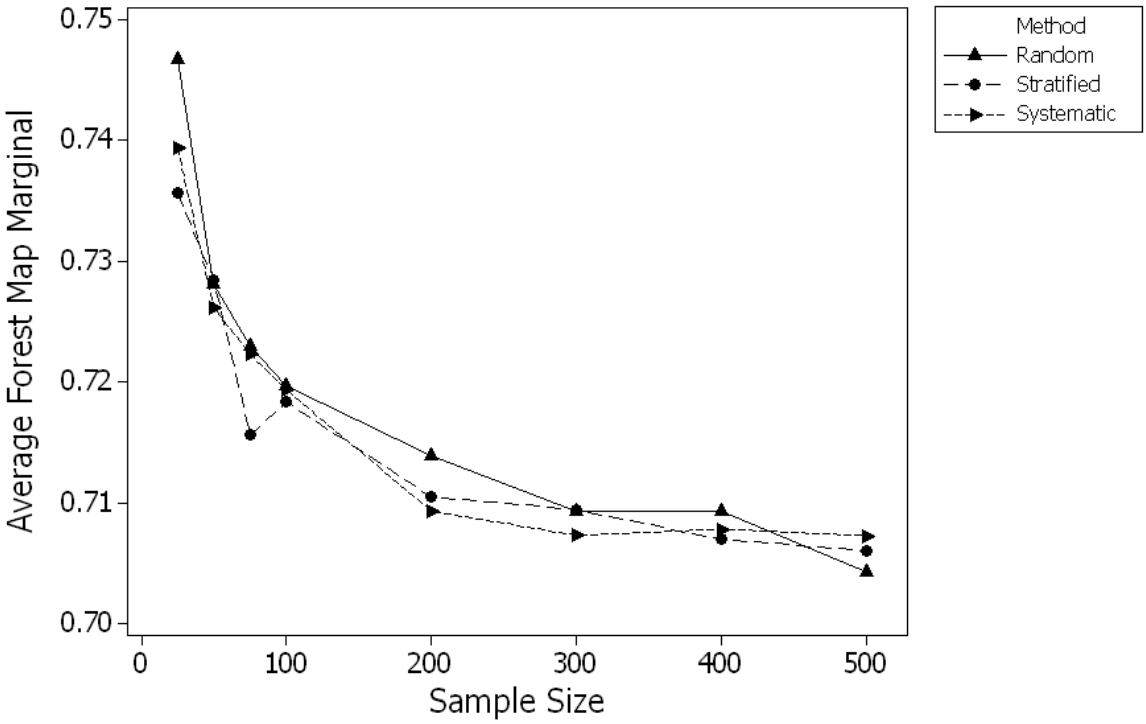


Figure 11a. Boxplot of Forest Map Marginals (Proportion) by Sampling Method and Sample Size for VA15

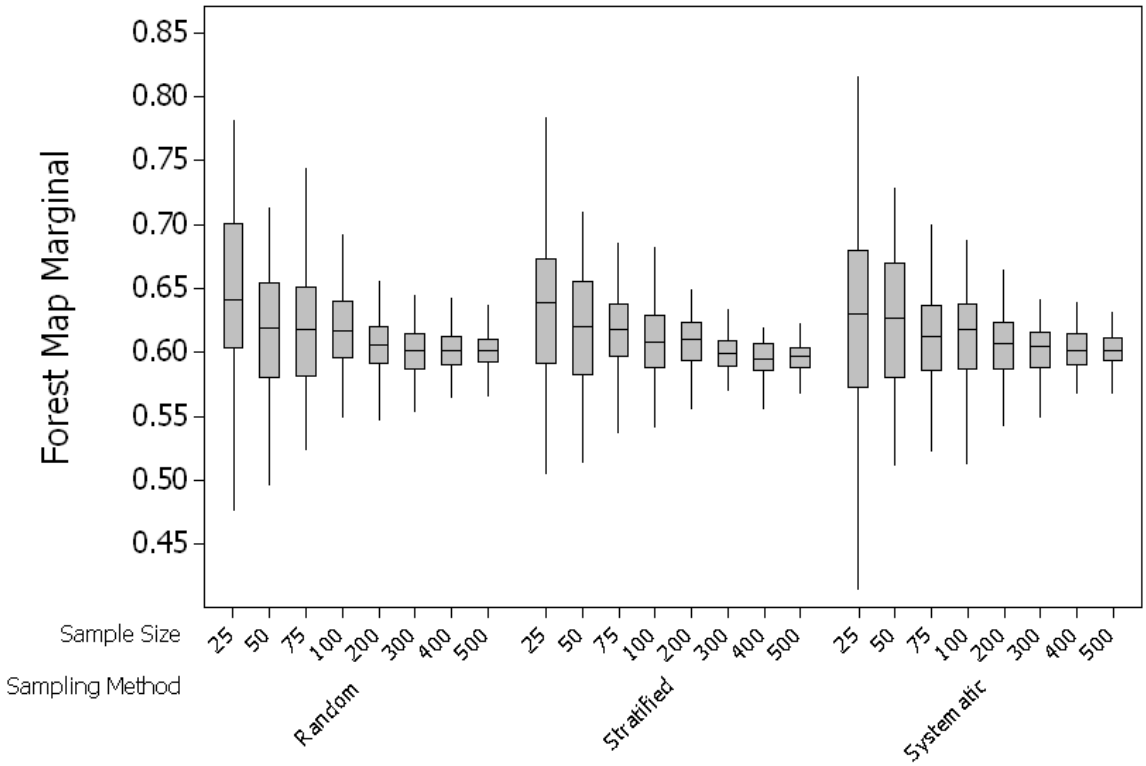


Figure 11b. Boxplot of Forest Map Marginals (Proportion) by Sampling Method and Sample Size for VA16

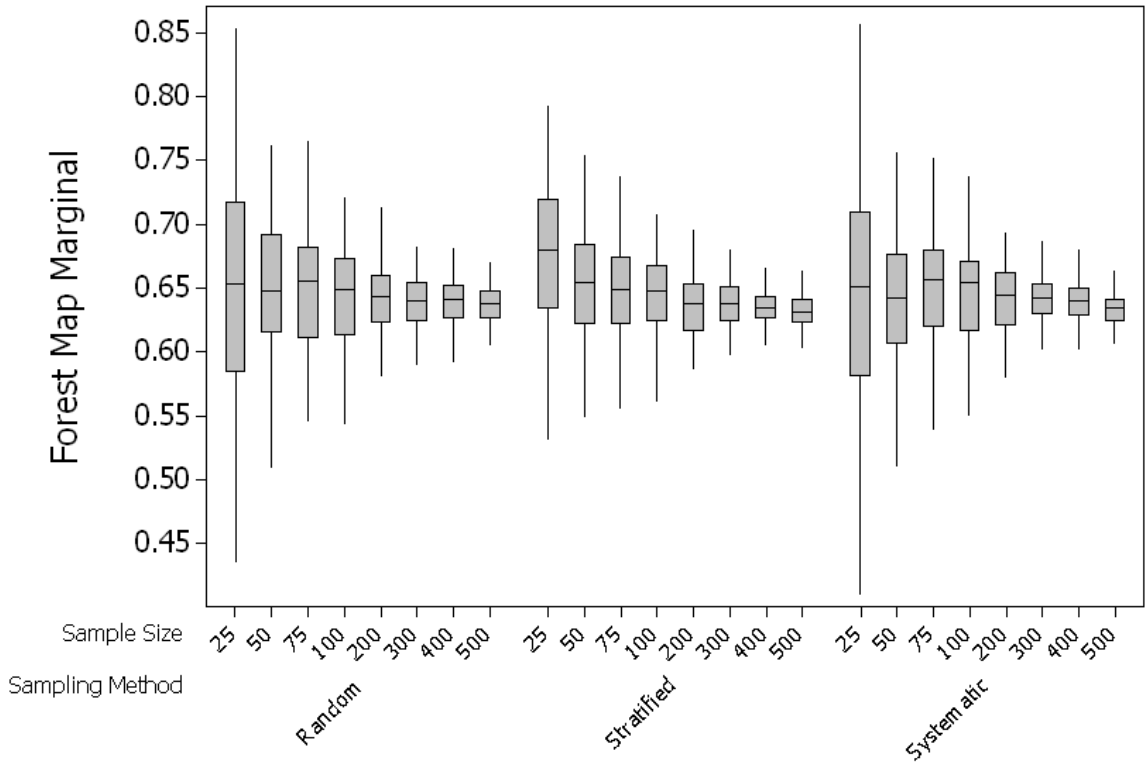


Figure 11c. Boxplot of Forest Map Marginals (Proportion) by Sampling Method and Sample Size for VA17

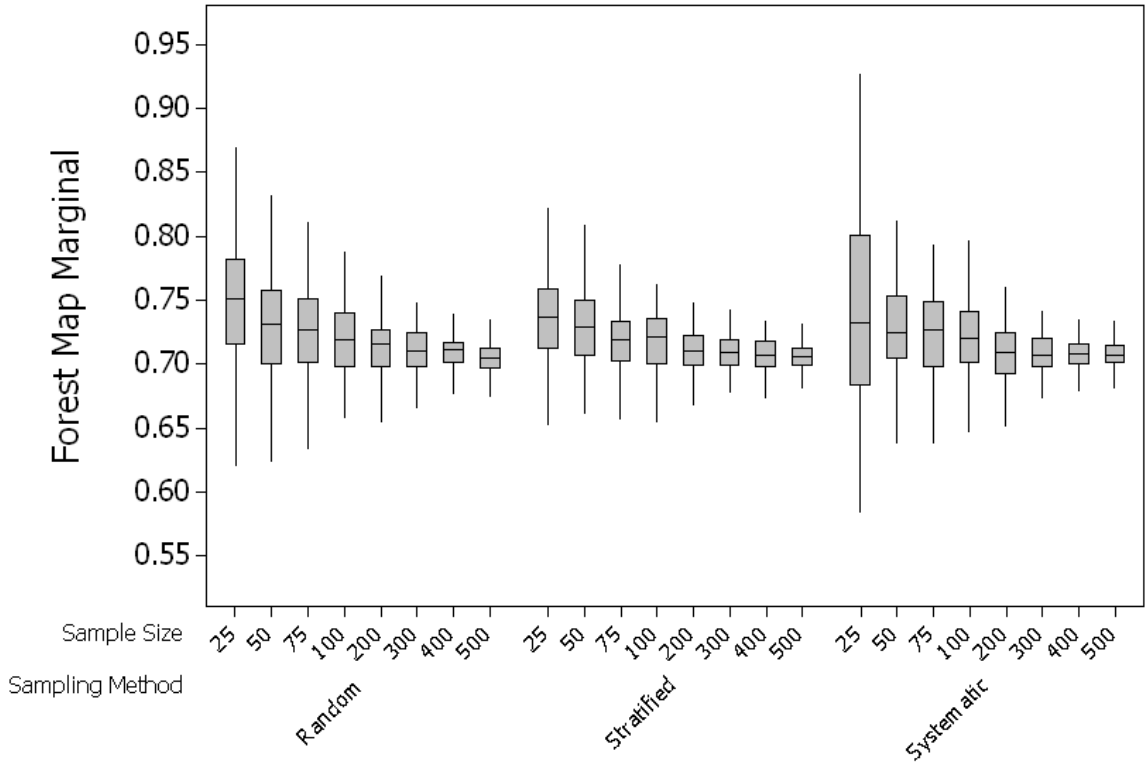


Figure 12a. Boxplot of Forest Area Estimates (Proportion) by Estimation Method and Sample Size for VA15 with Random Sampling

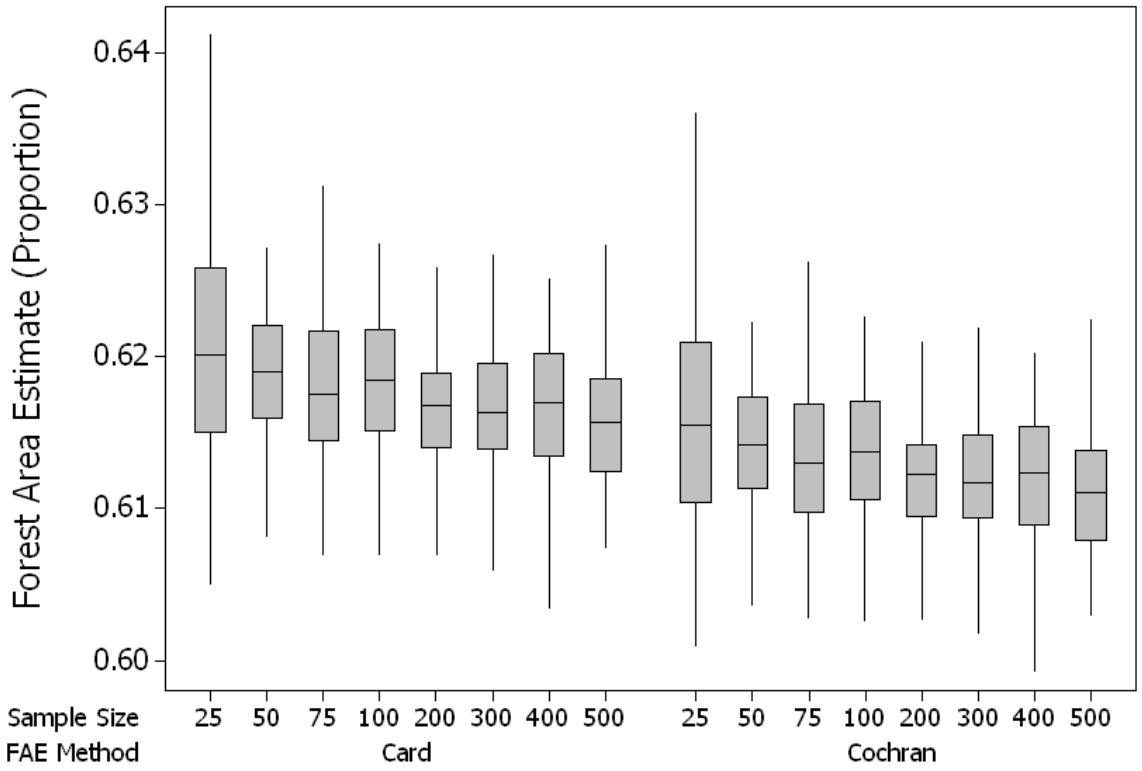


Figure 12b. Boxplot of Forest Area Estimates (Proportion) by Estimation Method and Sample Size for VA16 with Random Sampling

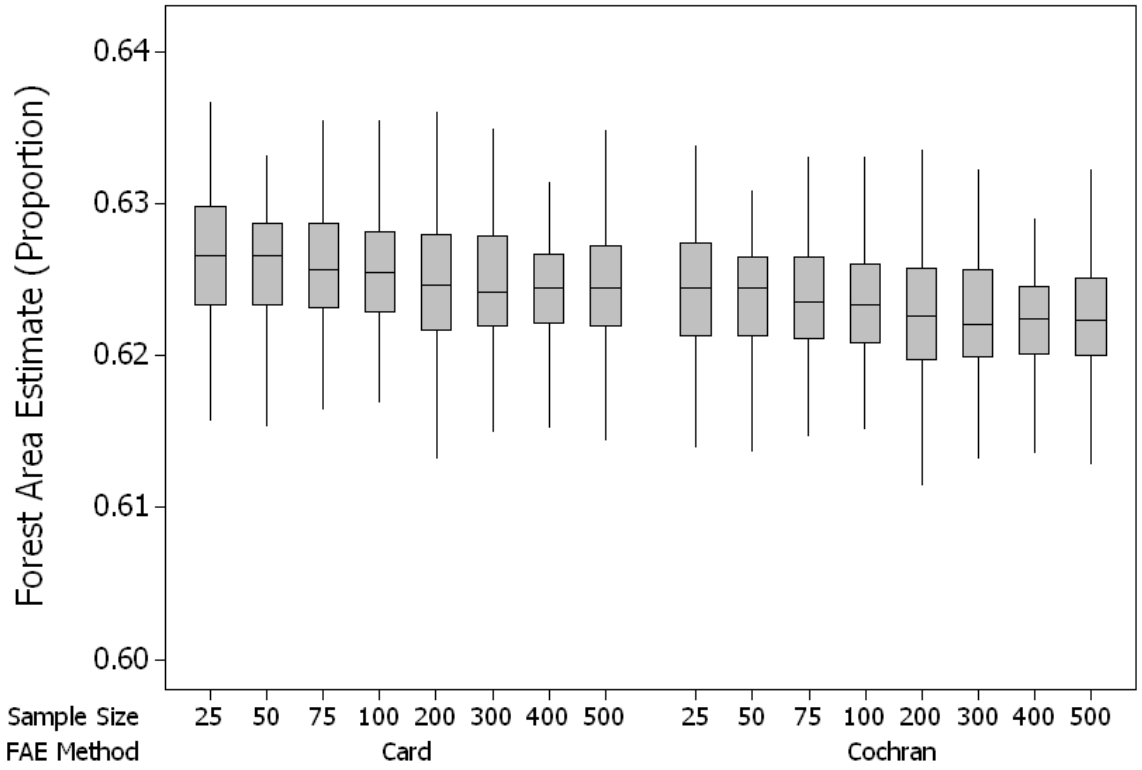


Figure 12c. Boxplot of Forest Area Estimates (Proportion) by Estimation Method and Sample Size for VA17 with Random Sampling

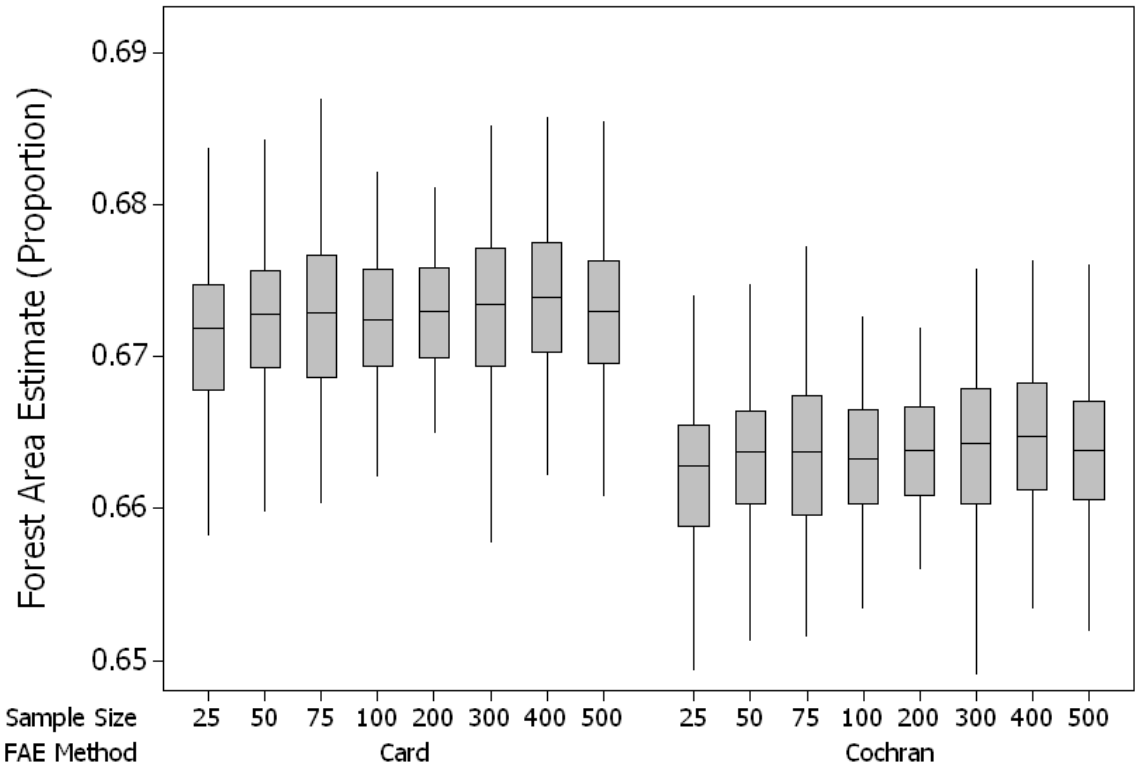


Figure 13a. VA15 Average Forest Area Estimate (Proportion) versus Sample Size by Estimation and Sampling Method

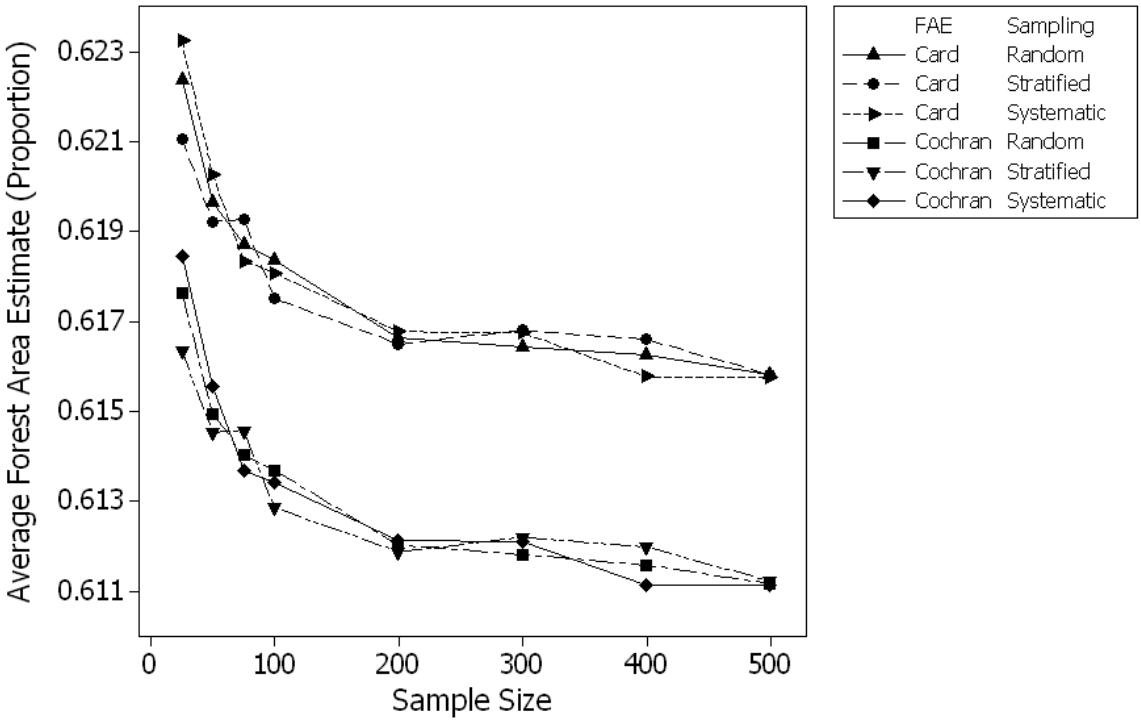


Figure 13b. VA16 Average Forest Area Estimate (Proportion) versus Sample Size by Estimation and Sampling Method

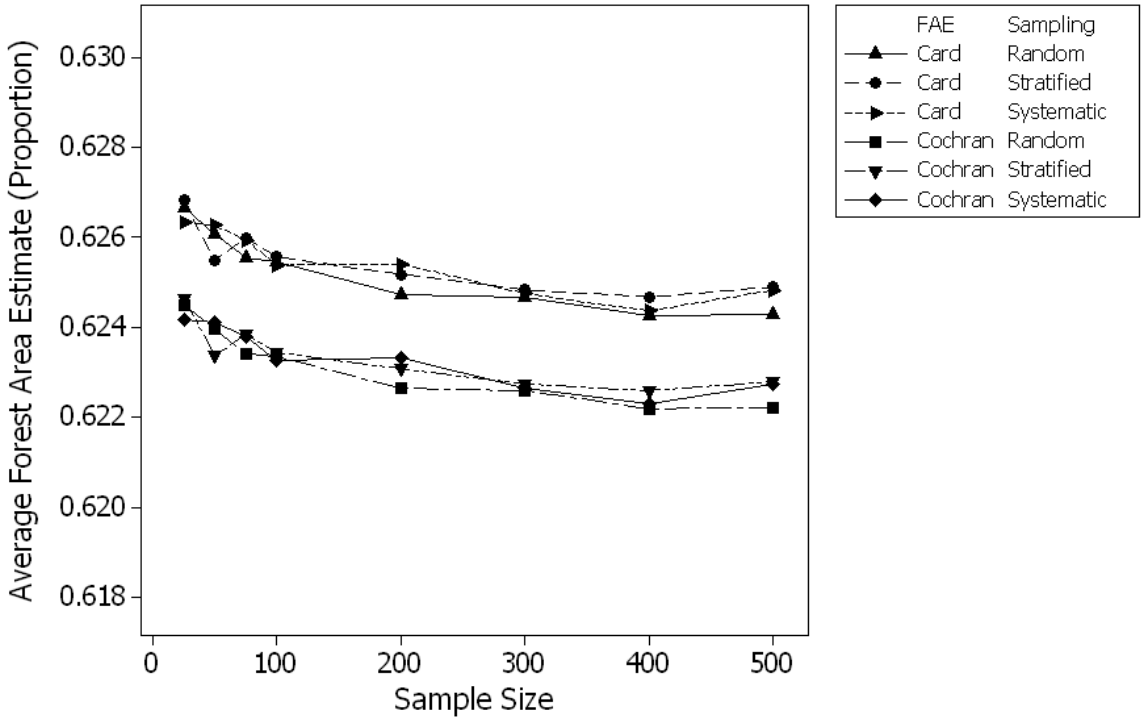


Figure 13c. VA17 Average Forest Area Estimate (Proportion) versus Sample Size by Estimation and Sampling Method

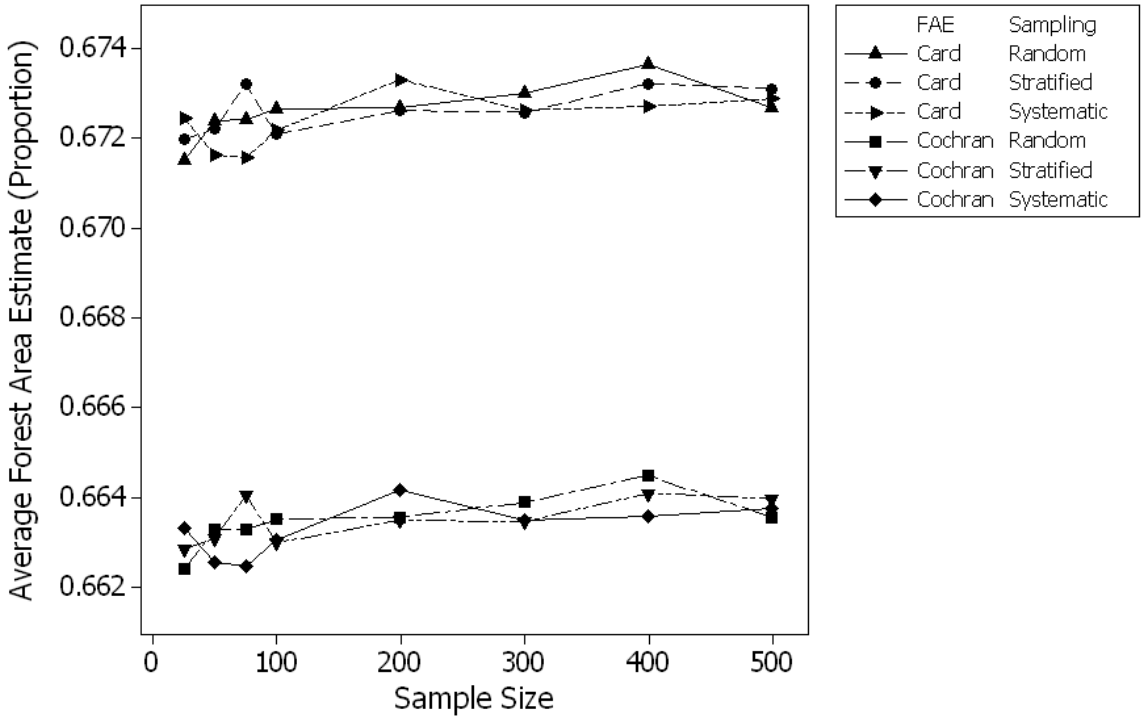


Figure 14a. Boxplot of the Precisions of Forest Area Estimates by Estimation Method and Sample Size for VA15 with Random Sampling

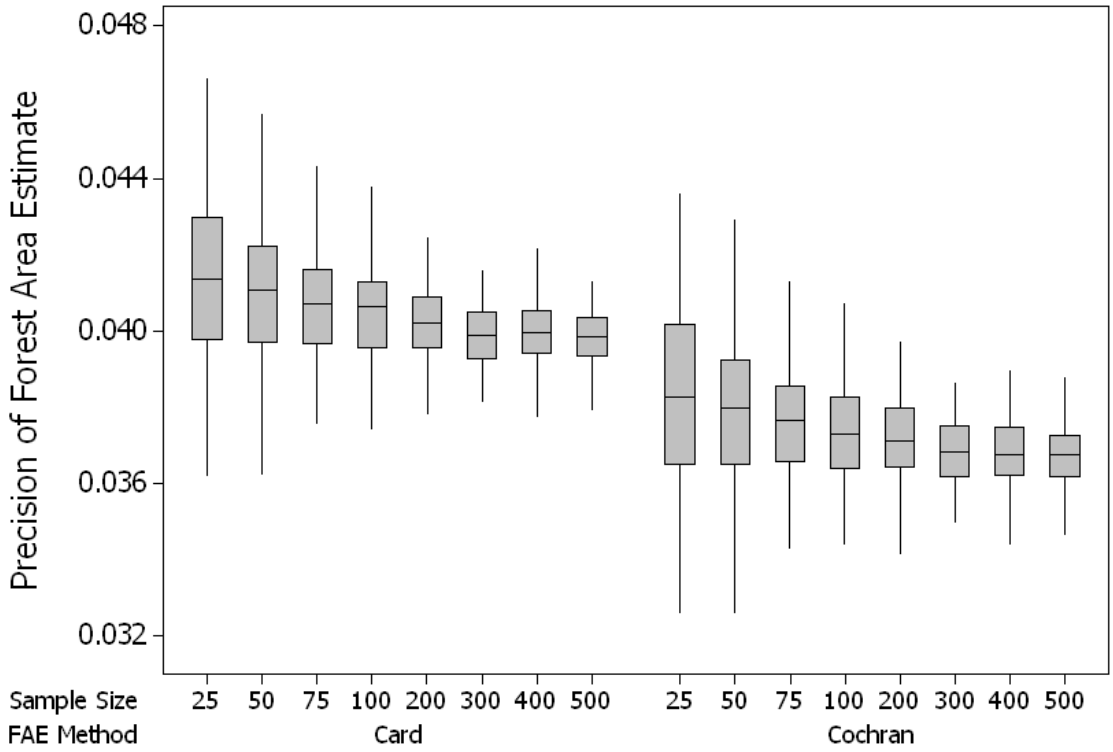


Figure 14b. Boxplot of the Precisions of Forest Area Estimates by Estimation Method and Sample Size for VA16 with Random Sampling

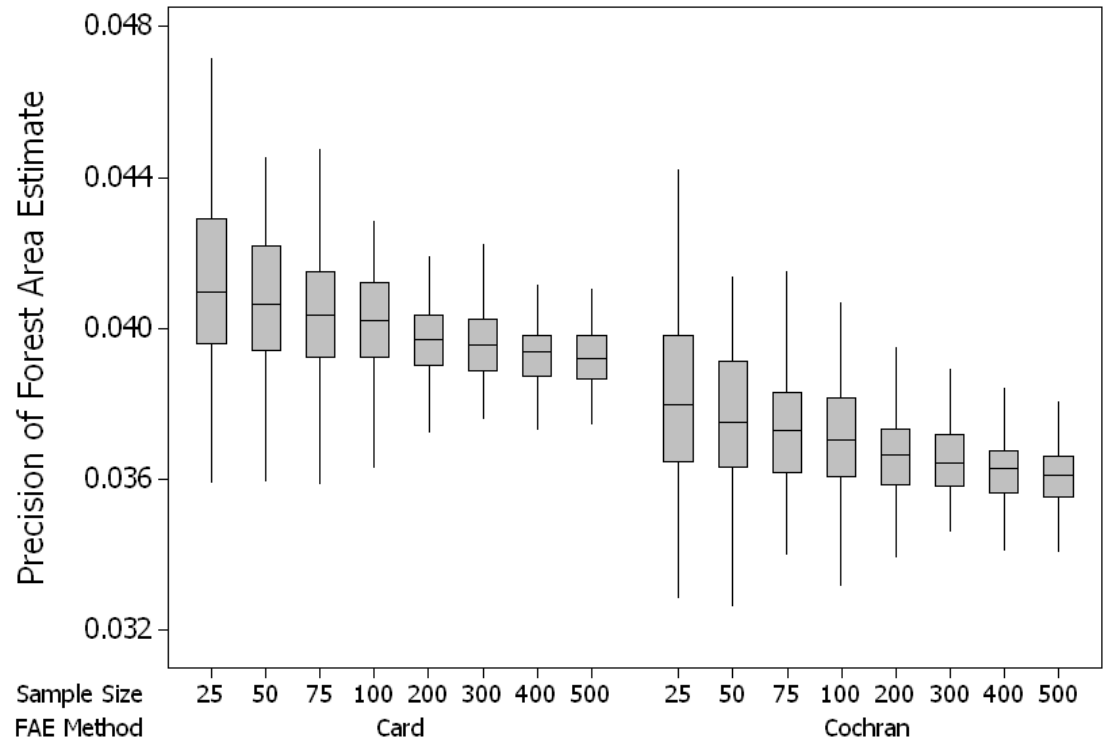
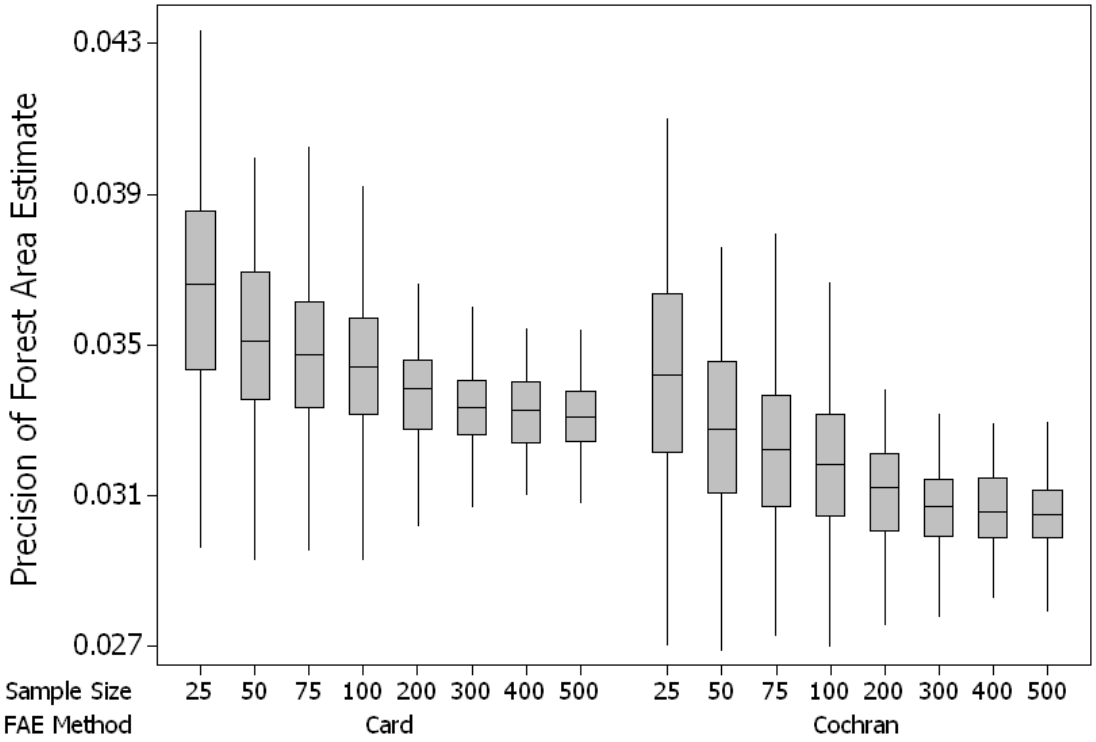


Figure 14c. Boxplot of the Precisions of Forest Area Estimates by Estimation Method and Sample Size for VA17 with Random Sampling



average classification accuracy for the seventy-two combinations was 5.45%, 5.31%, and 3.47%, respectively, for VA15, VA16, and VA17. A plot, for each image, of the average classification accuracy versus sample size by each combination of training data sample type (i.e., edited or unedited) and sampling method are in Figures 15a-15c.

3.4 Band Combinations

3.4.1 Classifications with All Six Bands: Are They the Most Accurate?

Although all six bands frequently produced the highest average accuracies with larger training data sample sizes for both VA15 and VA16 (Tables 10a-10b), on an individual draw basis, only 2,866 of the 63,925 total sample draws resulted in the highest classification accuracy when all six bands were used. More than half of the draws where all six bands produced the most accurate classification were from VA15. A majority of these draws were created with

Figure 15a. VA15 Average Classification Accuracy (%) versus Sample Size by Training Data Type and Sampling Method

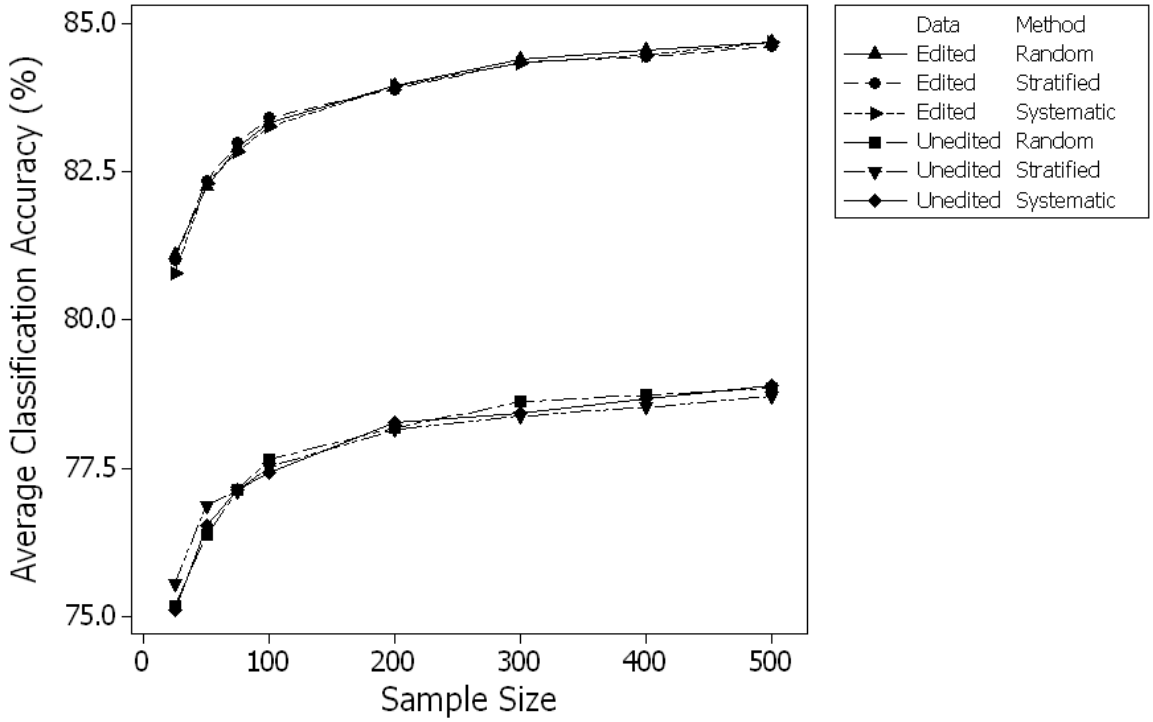


Figure 15b. VA16 Average Classification Accuracy (%) versus Sample Size by Training Data Type and Sampling Method

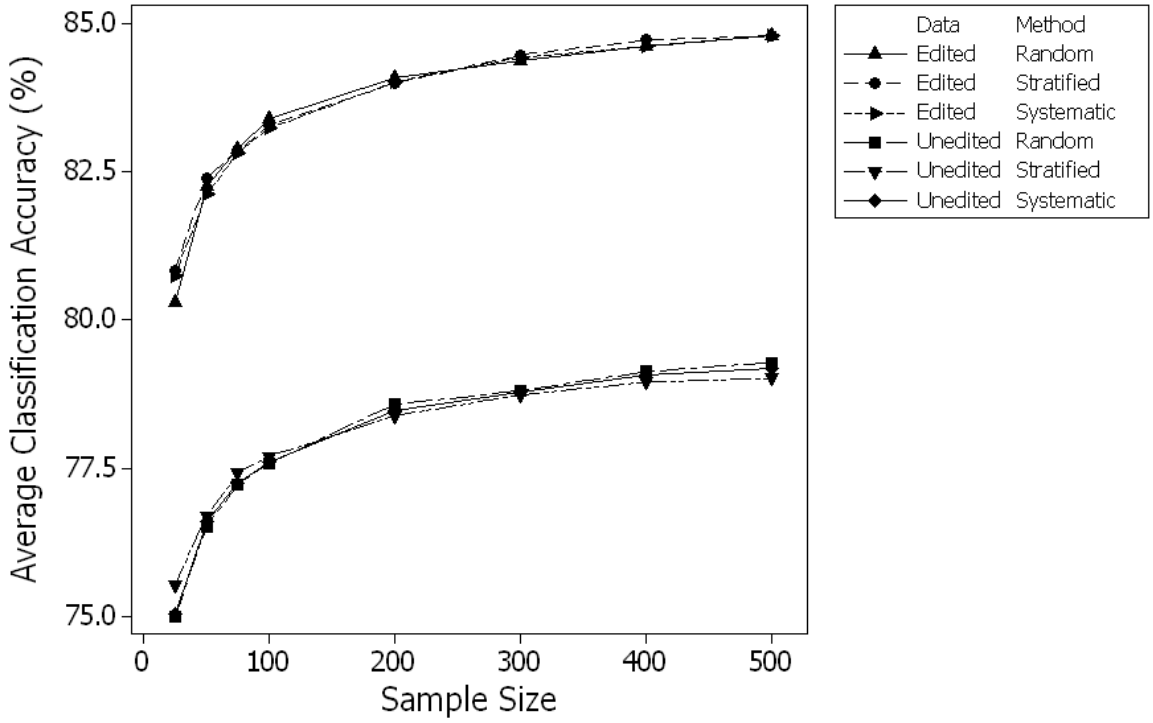
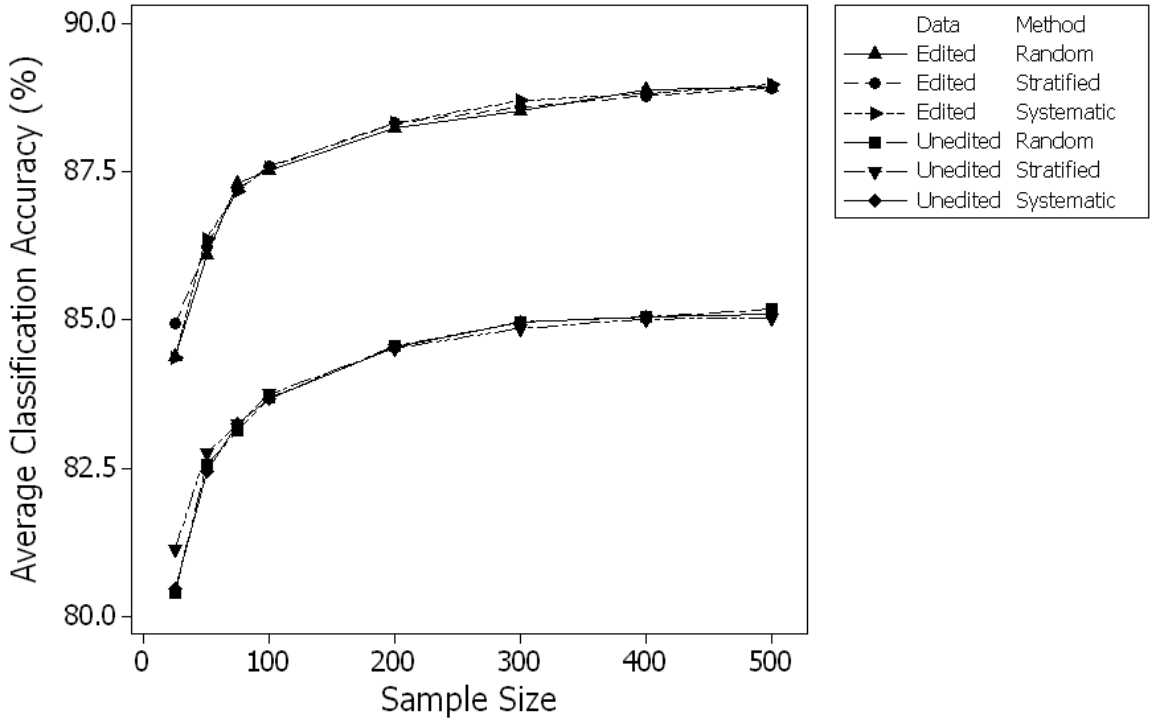


Figure 15c. VA17 Average Classification Accuracy (%) versus Sample Size by Training Data Type and Sampling Method



stratified random sampling and larger sample sizes. VA17 only required a maximum of four bands to produce the highest average accuracies (Table 10c). The average increase in classification accuracy by using the best band combination instead of all six bands decreases as the training data sample size increases, but still ranges between 0.48 and 1.20 percent, even for stratified random sampling with a sample size of 500 (Figures 16a-16c). Table 11 contains the band combinations that correspond to the band combination reference numbers in Figures 16a-16i.

When the band combinations which produced the most accurate classification for a given training data sample draw are summarized by image, sampling method, and sample size, an individual band combination was best a maximum of 28.1, 31.5, or 17.4 percent of the time for VA15, VA16, and VA17, respectively. The characteristics of a given sample draw, which are

clearly influenced by the image, sample size, and sampling method, appear to have the greatest impact on which band combination performs best.

3.4.1.1 Best Size of Band Combination for each Image, Sampling Method, and Sample Size Combination

The number of bands which produced the highest average accuracy for each training data sample size with VA15 was very similar across the three sampling methods. Only two sample sizes, 50 and 100, had a different number of bands with one sampling method than with the other two sampling methods for VA15 (Table 10a). In comparison, VA16 only had one sample size, 200, and VA17 did not have any sample sizes, where all three sampling methods produced the highest average accuracy with the same number of bands (Tables 10b-10c). All six bands never produced the highest average accuracy with any combination of sampling method and sample size for VA17.

The sampling method influenced the relationship between the training data sample size and the number of bands that produces the highest average accuracy. Stratified random samples required more bands at smaller training data sample sizes, especially with VA16 and VA17 (Tables 10a-10c). These findings indicate that the number of Landsat ETM+ bands that will result in the highest accuracies are influenced by the particular image, the size of the training data sample, and, to a lesser degree, the sampling method used to obtain the training data.

3.4.1.2 Top Band Combinations

3.4.1.2.1 Trends Among All Band Combinations

A similar overall pattern in terms of the variation in average accuracies among the sixty-three possible band combinations was observed across all sample sizes for each of the nine image and sampling method pairs (Figures 17a-17i, single band combinations were excluded

from these figures). Only slight variations were observed in the trends of average accuracy for all band combinations with different sampling methods. Variations between the methods were smallest with VA17 and greatest for VA15. In all cases, random and systematic samples had almost identical trends. Stratified random samples were occasionally a little different, especially when fewer bands were considered. The differences between sampling methods tended to decrease as more bands were considered, especially for VA15 and VA16.

3.4.1.2.2 Top Band Combinations by Number of Bands

The average classification accuracy of all classifications with band combinations of a given size tended to increase as the number of bands increased, but leveled off between three and six bands depending on the image, sampling method, and sample size. This is illustrated by plots of the largest average classification accuracy from a band combination of a given size (i.e., 1 through 6) versus the number of bands by training data sample size (Figures 18a-18i). In general, as the sample size increased, more bands produced higher average accuracies. The Hughes effect is especially evident with small training data sample sizes in Figure 18d, where the average accuracy of the best band combination decreased after two or three bands as the number of bands increased. Table 12 contains a summary of the band combinations that were included in Figures 18a-18i. The third column in Table 12 is the proportion of the 72 factor combinations (image, sampling method, and sample size) that resulted in the maximum average accuracy with the given band combination of a specific size. The band combinations listed in Table 12 are likely to result in good discrimination between forest and nonforest, depending on the characteristics of the training data sample.

3.5 Band Selection Methods

The minimum and average Euclidian distances between pixels in a training data sample that belonged to different informational classes had very low correlations with classification accuracies (Table 13). These separability indices were not useful in selecting the best band combination for a given training data sample with any of the three images. Low correlations were also obtained between the leave-one-out cross-validation accuracies and the classification accuracies (Table 13). A strong relationship did appear between the two accuracies when samples of a single sample size, 500, were considered (Table 13 and Figures 19-20). Figure 21 contains a plot with both the average cross-validation accuracy and the average classification accuracy versus sample size for random samples with VA15. Despite the high correlations for sample sizes of 500, the band combinations that corresponded with the highest cross-validation accuracies resulted in classifications that, on average, were less accurate than classifications with all six bands. However, cross-validation accuracies with a sufficiently large sample size could be used as an indicator of training data sample quality.

Table 10a. VA15 Number of Bands and Band Combination with Highest Average Accuracy for Each Combination of Sample Size and Sampling Method

Image	Sampling Method	Sample Size	Number of Bands	Average Accuracy	Band Combination					
15	Random	25	3	75.95	1	2		4		
15	Random	50	3	76.77	1	2		4		
15	Random	75	5	77.24	1	2	3	4		6
15	Random	100	6	77.65	1	2	3	4	5	6
15	Random	200	6	78.18	1	2	3	4	5	6
15	Random	300	6	78.62	1	2	3	4	5	6
15	Random	400	6	78.73	1	2	3	4	5	6
15	Random	500	6	78.85	1	2	3	4	5	6
15	Stratified	25	3	76.36	1	2		4		
15	Stratified	50	5	77.00	1	2	3	4		6
15	Stratified	75	5	77.20	1	2	3	4		6
15	Stratified	100	6	77.51	1	2	3	4	5	6
15	Stratified	200	6	78.15	1	2	3	4	5	6
15	Stratified	300	6	78.38	1	2	3	4	5	6
15	Stratified	400	6	78.53	1	2	3	4	5	6
15	Stratified	500	6	78.71	1	2	3	4	5	6
15	Systematic	25	3	75.96	1	2		4		
15	Systematic	50	3	76.86	1	2		4		
15	Systematic	75	5	77.23	1	2	3	4		6
15	Systematic	100	5	77.51	1	2	3	4		6
15	Systematic	200	6	78.26	1	2	3	4	5	6
15	Systematic	300	6	78.44	1	2	3	4	5	6
15	Systematic	400	6	78.68	1	2	3	4	5	6
15	Systematic	500	6	78.90	1	2	3	4	5	6

Table 10b. VA16 Number of Bands and Band Combination with Highest Average Accuracy for Each Combination of Sample Size and Sampling Method

Image	Sampling Method	Sample Size	Number of Bands	Average Accuracy	Band Combination
16	Random	25	1	77.02	2
16	Random	50	3	77.37	1 2 6
16	Random	75	3	77.91	1 2 6
16	Random	100	3	78.11	1 2 6
16	Random	200	5	78.71	1 2 3 4 6
16	Random	300	5	78.96	1 2 3 4 6
16	Random	400	5	79.17	1 2 3 4 6
16	Random	500	6	79.28	1 2 3 4 5 6
16	Stratified	25	3	76.50	1 2 4
16	Stratified	50	4	77.03	1 2 3 4
16	Stratified	75	5	77.58	1 2 3 4 6
16	Stratified	100	5	77.79	1 2 3 4 5
16	Stratified	200	5	78.42	1 2 3 4 5
16	Stratified	300	6	78.74	1 2 3 4 5 6
16	Stratified	400	6	78.95	1 2 3 4 5 6
16	Stratified	500	6	79.02	1 2 3 4 5 6
16	Systematic	25	2	76.58	2
16	Systematic	50	3	77.33	1 2 6
16	Systematic	75	3	77.77	1 2 6
16	Systematic	100	4	77.93	1 2 3 4
16	Systematic	200	5	78.58	1 2 3 4 6
16	Systematic	300	5	78.85	1 2 3 4 5
16	Systematic	400	5	79.12	1 2 3 4 5
16	Systematic	500	5	79.25	1 2 3 4 6

Table 10c. VA17 Number of Bands and Band Combination with Highest Average Accuracy for Each Combination of Sample Size and Sampling Method

Image	Sampling Method	Sample Size	Number of Bands	Average Accuracy	Band Combination				
17	Random	25	3	81.97	1	2		4	
17	Random	50	2	83.52		2			6
17	Random	75	2	83.91		2			6
17	Random	100	2	84.30		2			6
17	Random	200	4	84.95	1	2		4	6
17	Random	300	4	85.21	1	2		4	6
17	Random	400	4	85.31	1	2		4	6
17	Random	500	4	85.40	1	2		4	6
17	Stratified	25	4	82.41	1	2		4	6
17	Stratified	50	4	83.53	1	2		4	6
17	Stratified	75	4	83.77	1	2		4	6
17	Stratified	100	4	84.12	1	2		4	6
17	Stratified	200	4	84.70	1	2		4	6
17	Stratified	300	5	84.99	1	2	3	4	6
17	Stratified	400	5	85.13	1	2	3	4	6
17	Stratified	500	5	85.15	1	2	3	4	6
17	Systematic	25	2	82.04		2		4	
17	Systematic	50	2	83.36		2			6
17	Systematic	75	2	84.00		2			6
17	Systematic	100	2	84.31		2			6
17	Systematic	200	2	84.93		2			6
17	Systematic	300	4	85.32	1	2		4	6
17	Systematic	400	4	85.52	1	2		4	6
17	Systematic	500	4	85.58	1	2		4	6

Table 11. Reference Numbers used in Figures for Each Band Combination

Band Combination Reference Number	Band Combination						Band Combination Reference Number	Band Combination					
1	1						33		2	3		5	
2		2					34		2	3			6
3			3				35		2		4	5	
4				4			36		2		4		6
5					5		37		2			5	6
6						6	38			3	4	5	
7	1	2					39			3	4		6
8	1		3				40			3		5	6
9	1			4			41				4	5	6
10	1				5		42	1	2	3	4		
11	1					6	43	1	2	3		5	
12		2	3				44	1	2	3			6
13		2		4			45	1	2		4	5	
14		2			5		46	1	2		4		6
15		2				6	47	1	2			5	6
16			3	4			48	1		3	4	5	
17			3		5		49	1		3	4		6
18			3			6	50	1		3		5	6
19				4	5		51	1			4	5	6
20				4		6	52		2	3	4	5	
21					5	6	53		2	3	4		6
22	1	2	3				54		2	3		5	6
23	1	2		4			55		2		4	5	6
24	1	2			5		56			3	4	5	6
25	1	2				6	57	1	2	3	4	5	
26	1		3	4			58	1	2	3	4		6
27	1		3		5		59	1	2	3		5	6
28	1		3			6	60	1	2		4	5	6
29	1			4	5		61	1		3	4	5	6
30	1			4		6	62		2	3	4	5	6
31	1				5	6	63	1	2	3	4	5	6
32		2	3	4									

Figure 16a. VA15 Average Increase in Classification Accuracy (%) (over using all 6 bands) with Best Band Combination versus Sample Size by Sampling Method

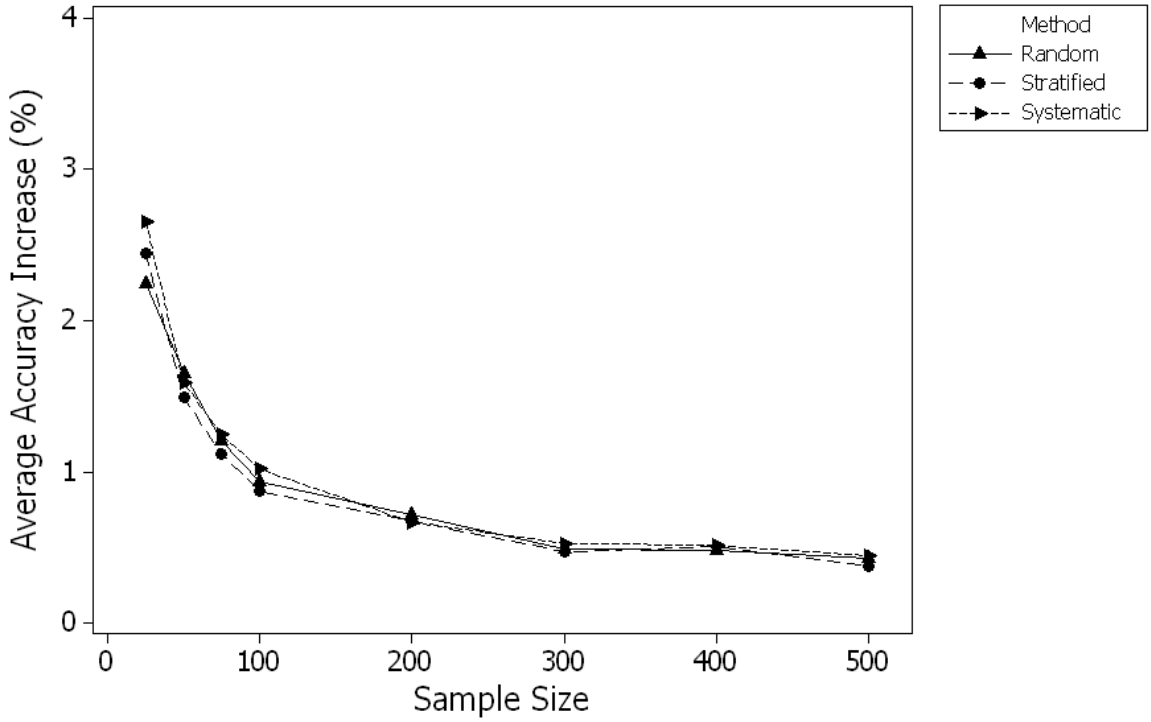


Figure 16b. VA16 Average Increase in Classification Accuracy (%) (over using all 6 bands) with Best Band Combination versus Sample Size by Sampling Method

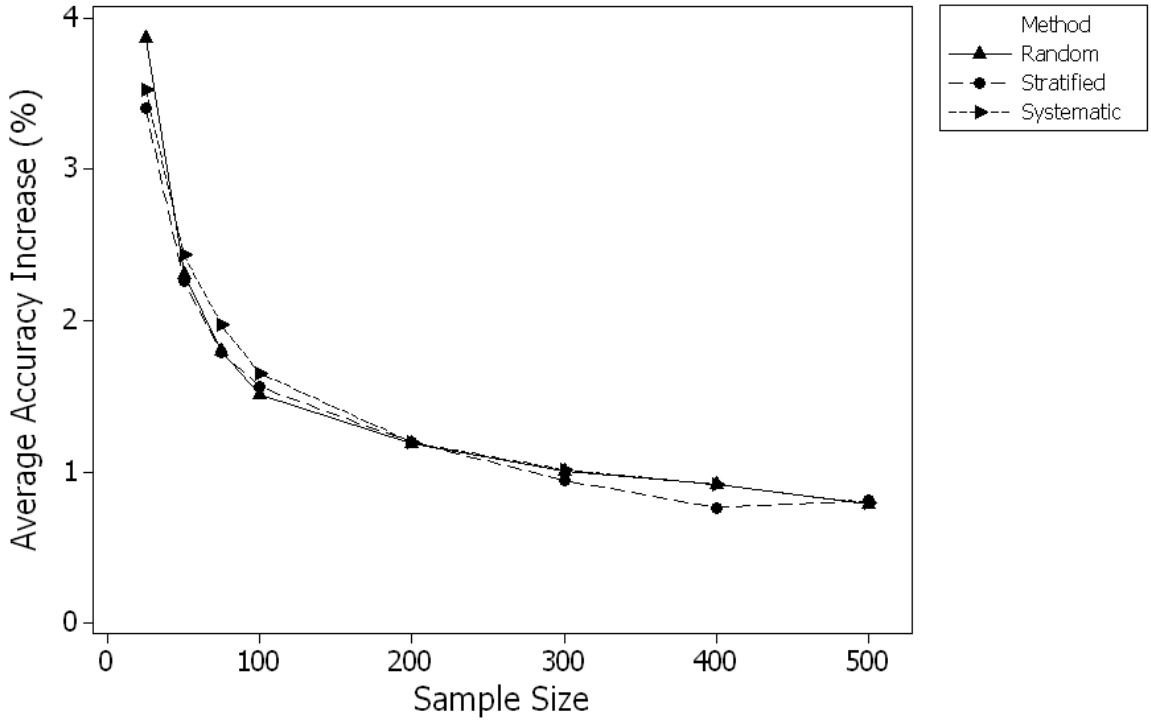


Figure 16c. VA17 Average Increase in Classification Accuracy (%) (over using all 6 bands) with Best Band Combination versus Sample Size by Sampling Method

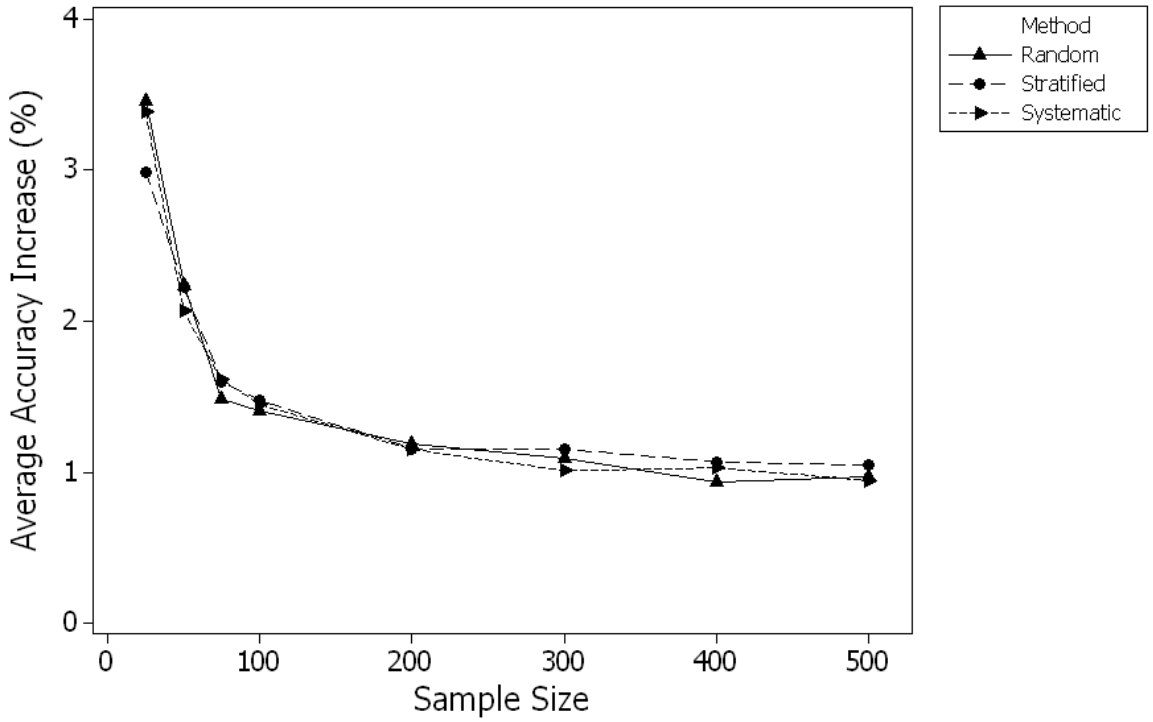
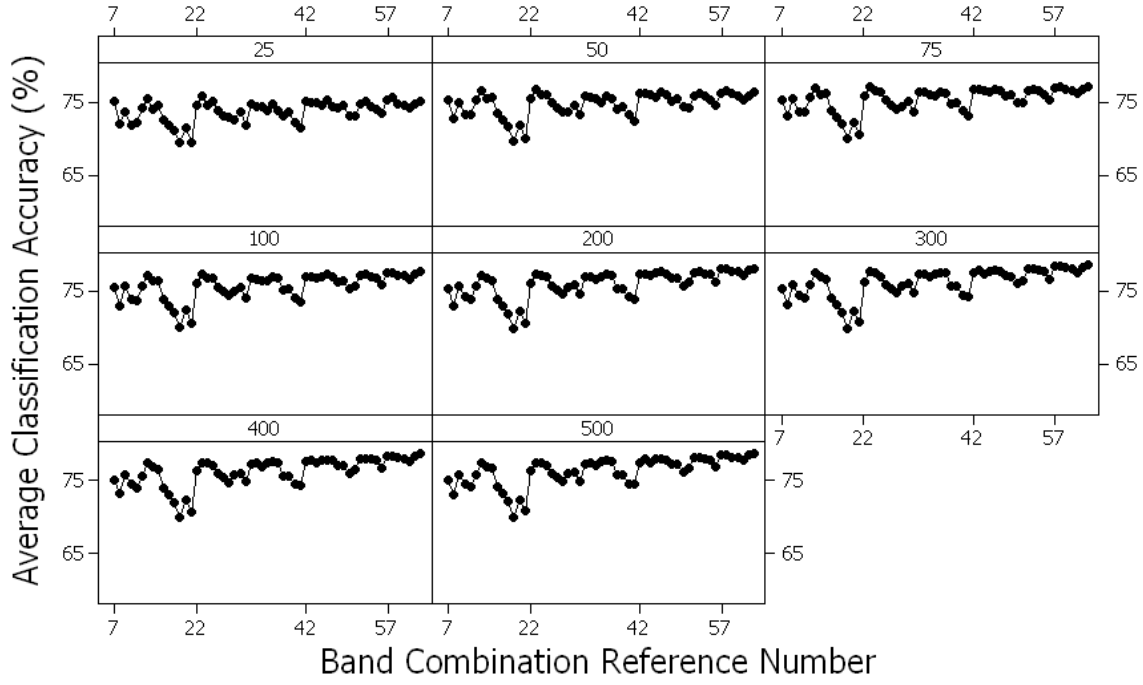


Figure 17a. VA15 Average Classification Accuracy (%) versus Band Combination by Sample Size with Random Sampling



Panel variable: Sample Size

Figure 17b. VA15 Average Classification Accuracy (%) versus Band Combination by Sample Size with Stratified Random Sampling

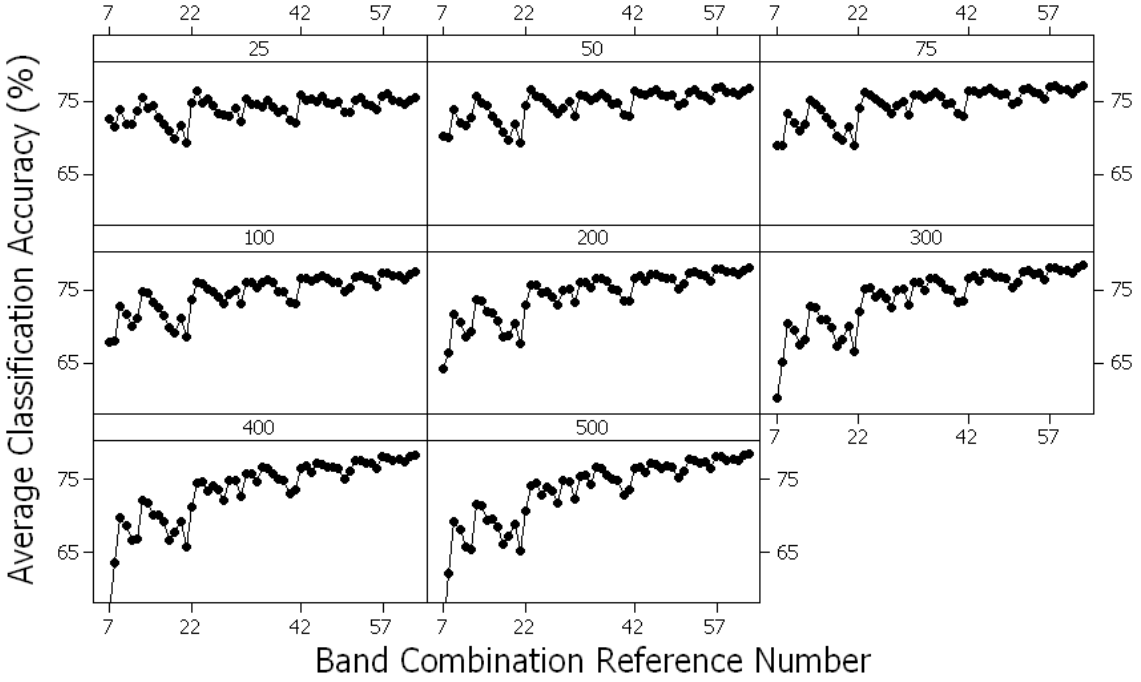


Figure 17c. VA15 Average Classification Accuracy (%) versus Band Combination by Sample Size with Systematic Sampling

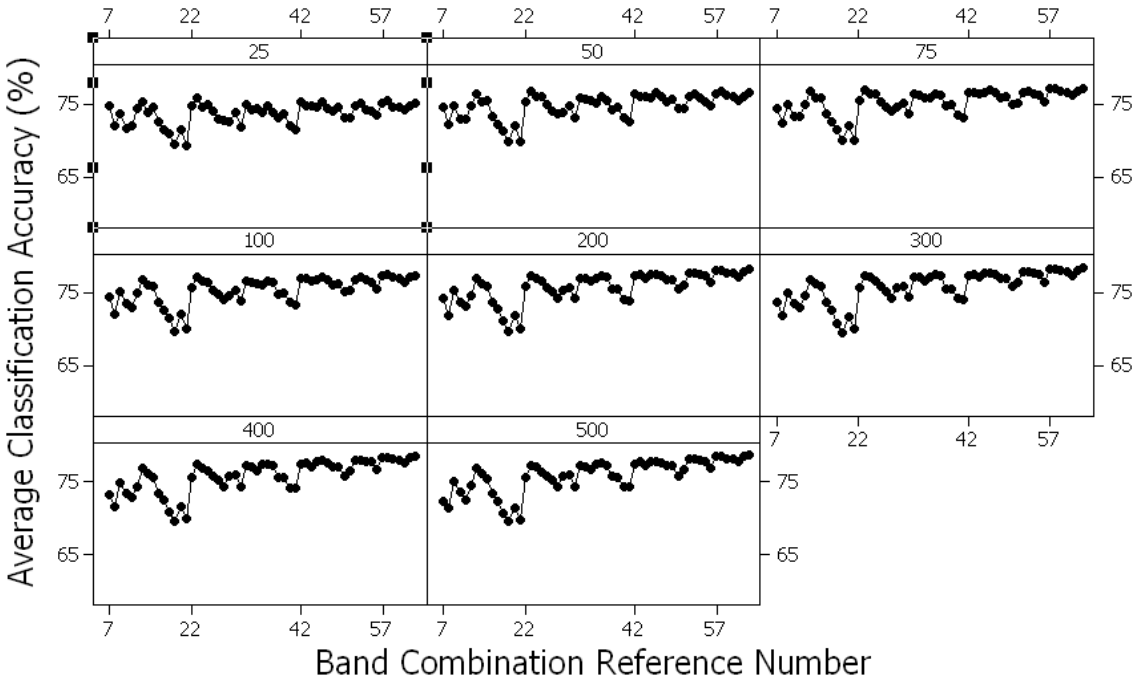


Figure 17d. VA16 Average Classification Accuracy (%) versus Band Combination by Sample Size with Random Sampling

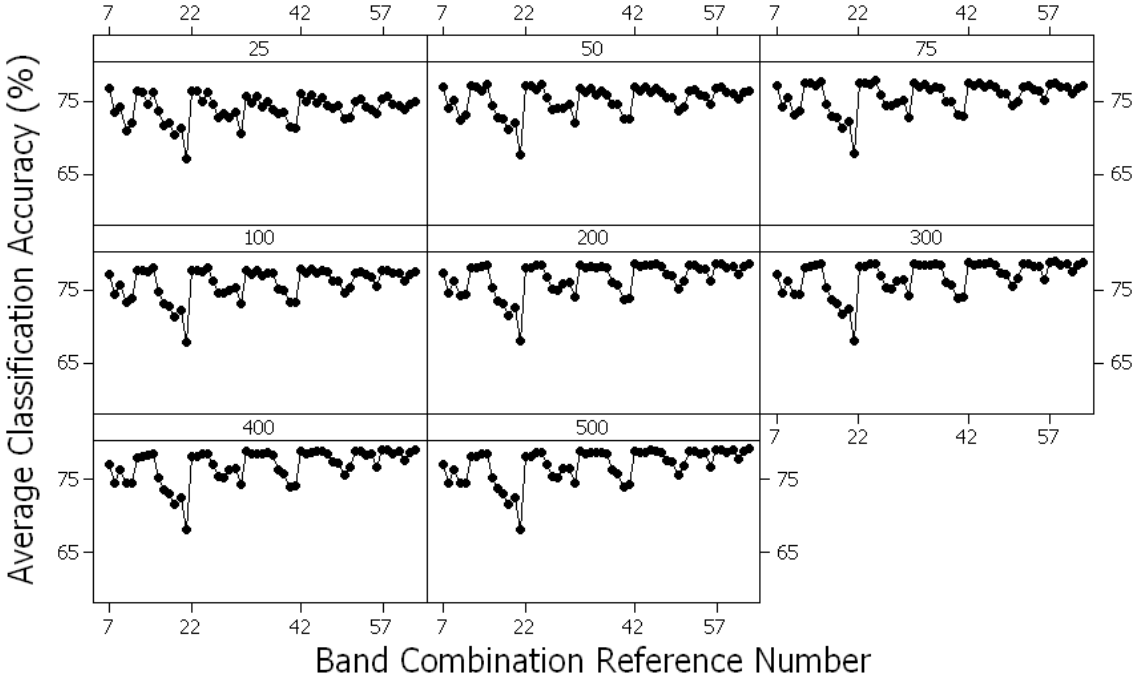


Figure 17e. VA16 Average Classification Accuracy (%) versus Band Combination by Sample Size with Stratified Random Sampling

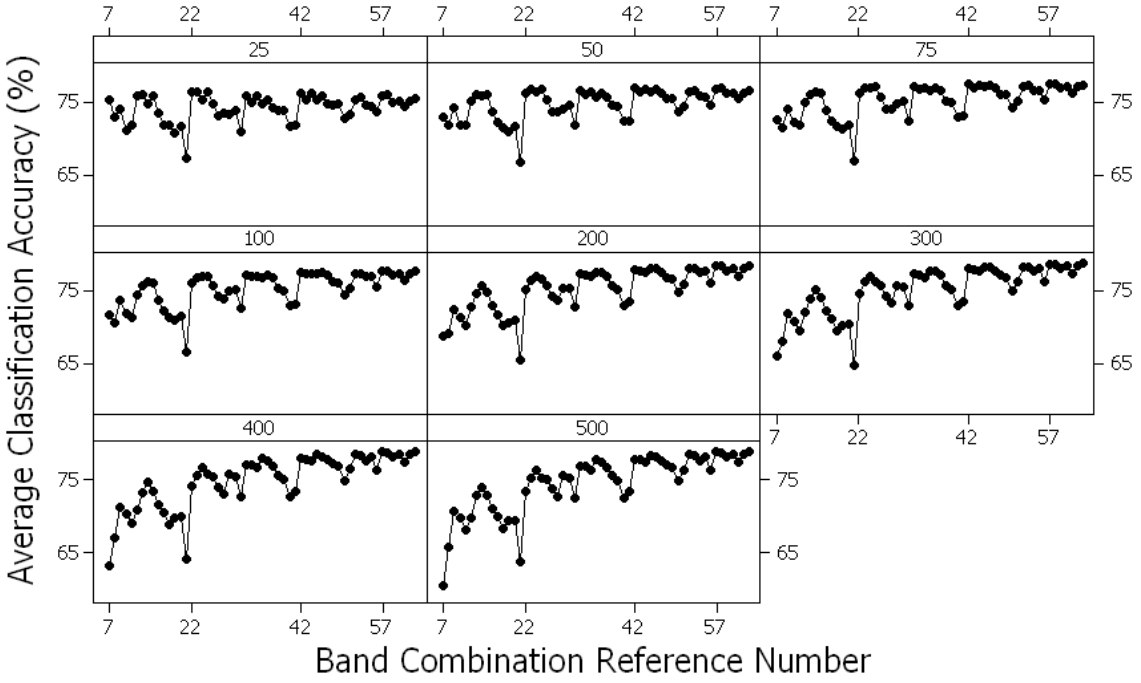


Figure 17f. VA16 Average Classification Accuracy (%) versus Band Combination by Sample Size with Systematic Sampling

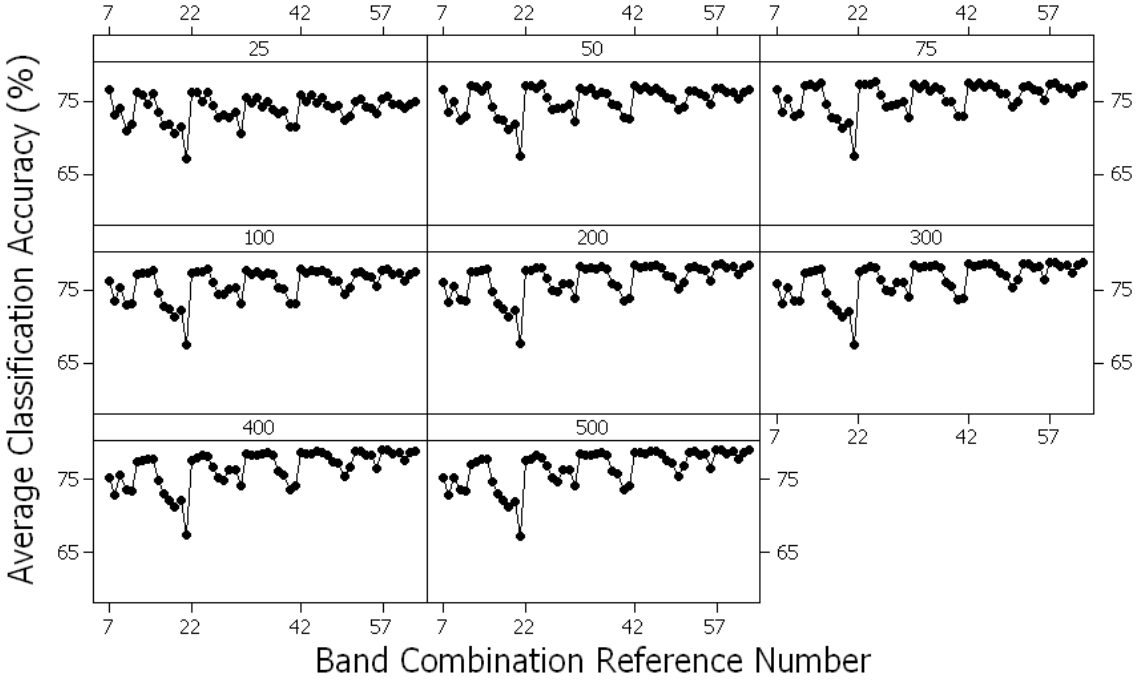


Figure 17g. VA17 Average Classification Accuracy (%) versus Band Combination by Sample Size with Random Sampling

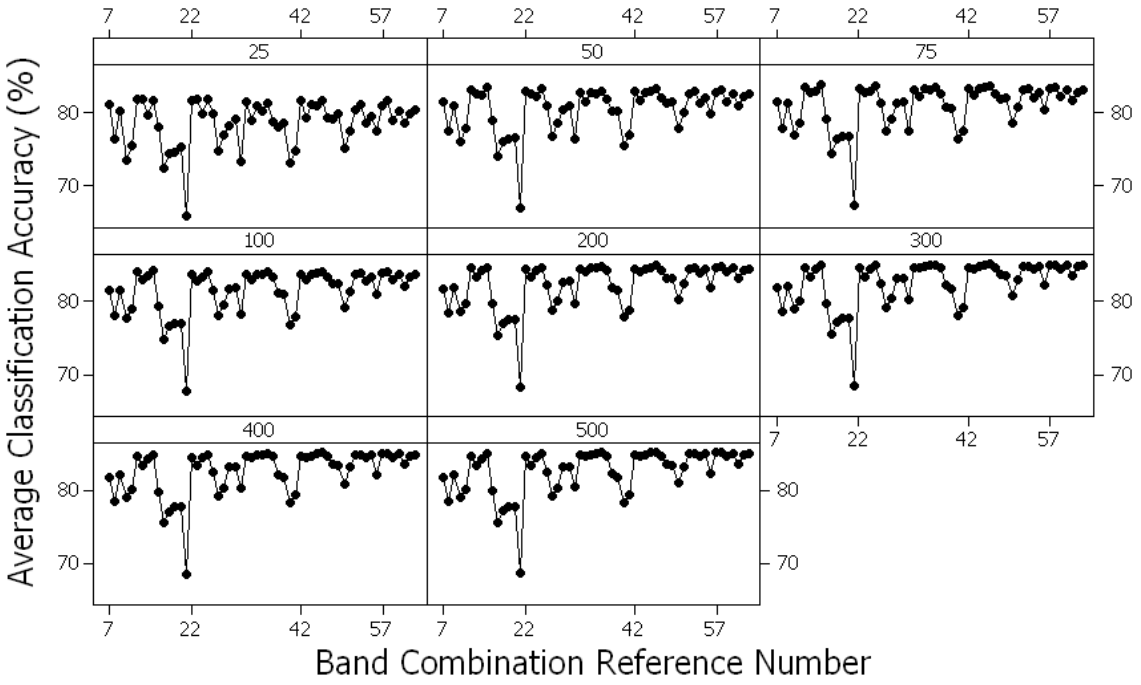


Figure 17h. VA17 Average Classification Accuracy (%) versus Band Combination by Sample Size with Stratified Random Sampling

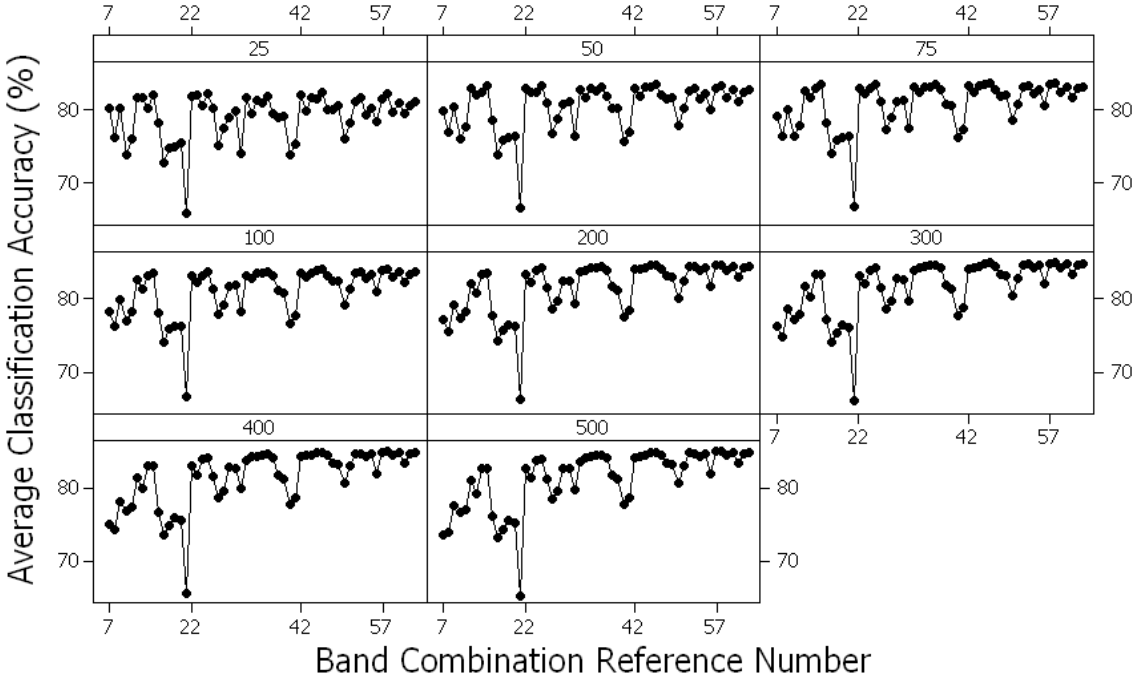


Figure 17i. VA17 Average Classification Accuracy (%) versus Band Combination by Sample Size with Systematic Sampling

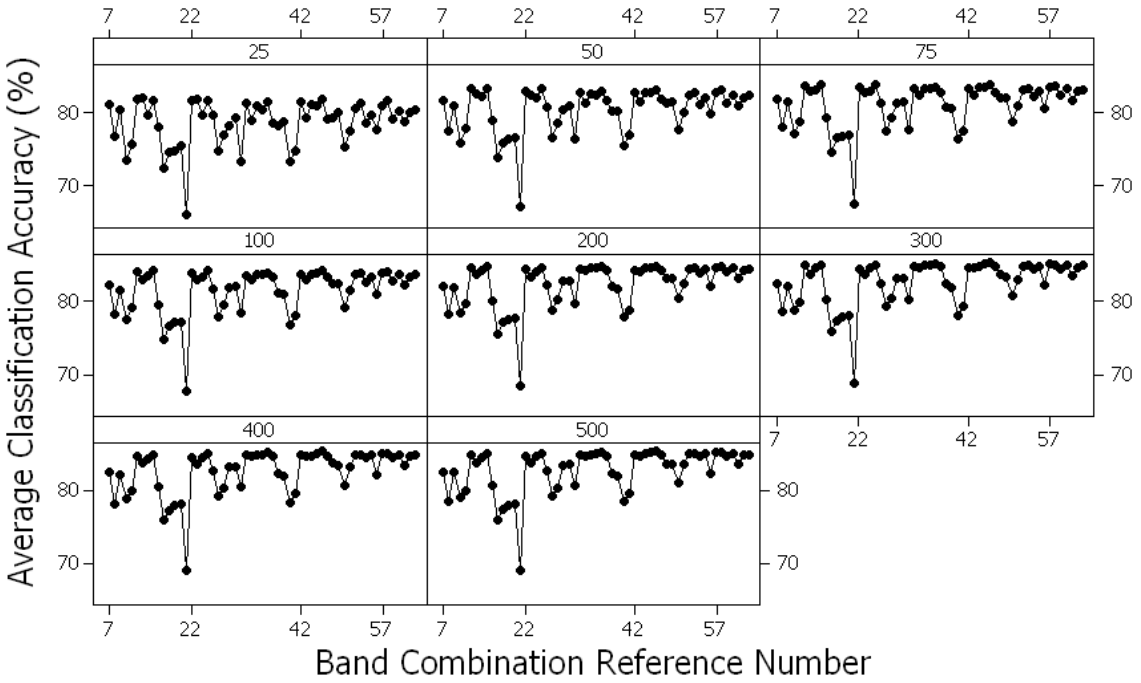


Figure 18a. VA15 Maximum Average Accuracy (%) versus Number of Bands by Sample Size with Random Sampling

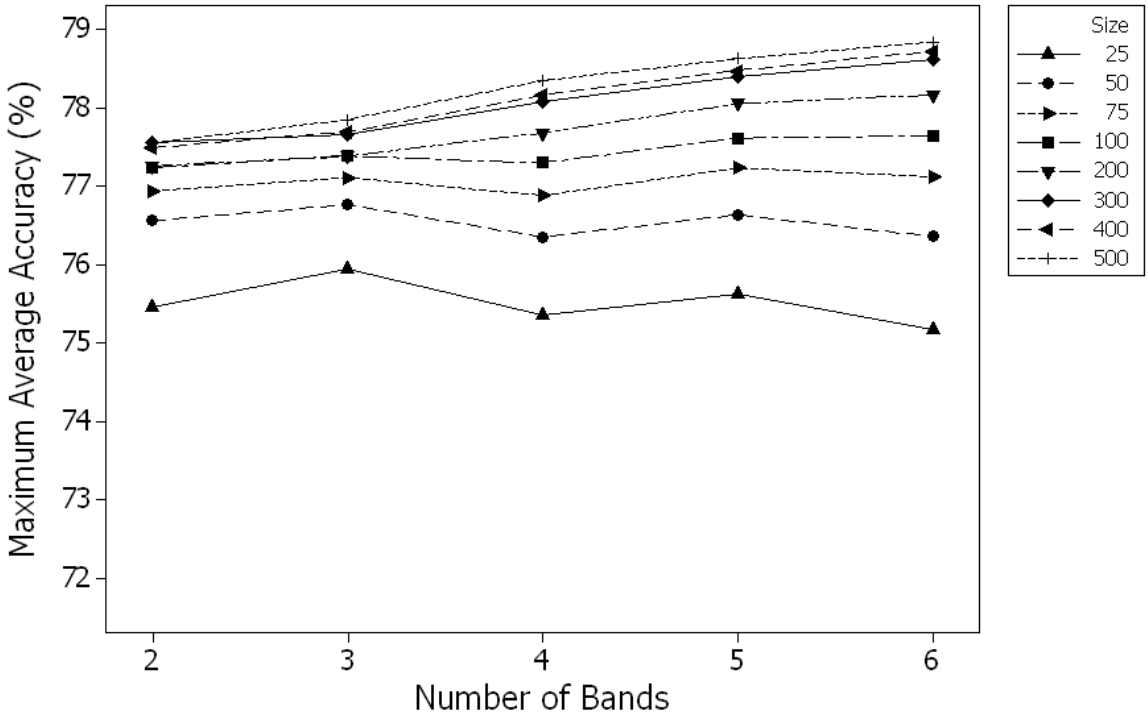


Figure 18b. VA15 Maximum Average Accuracy (%) versus Number of Bands by Sample Size with Stratified Random Sampling

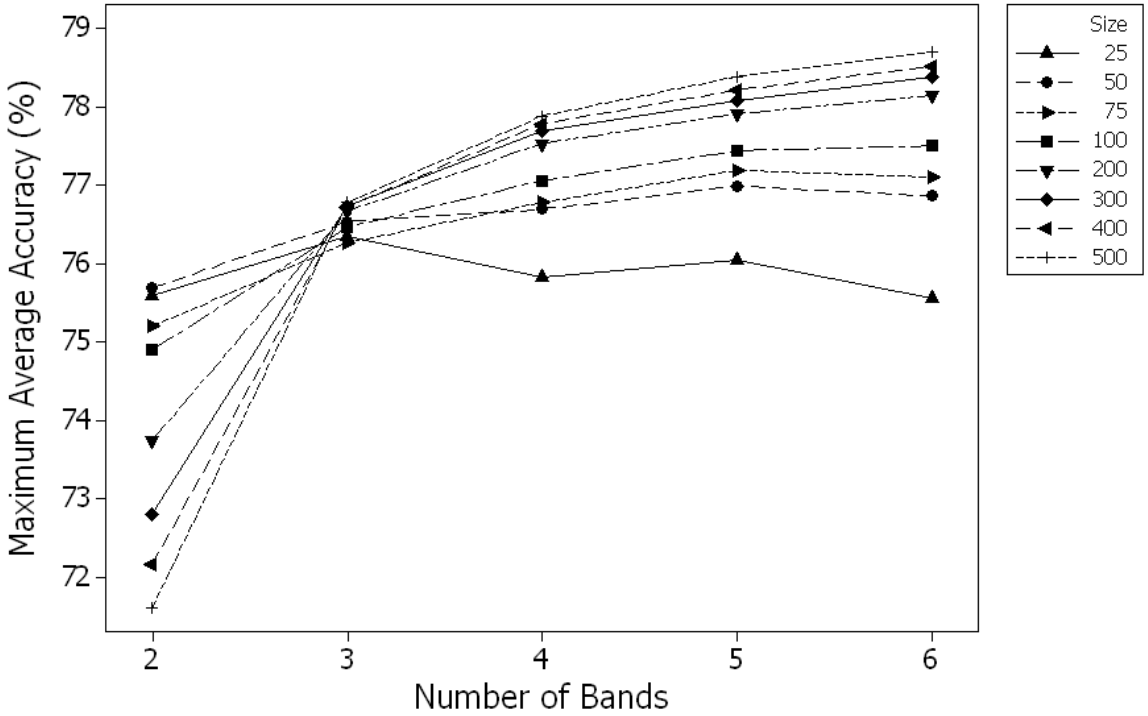


Figure 18c. VA15 Maximum Average Accuracy (%) versus Number of Bands by Sample Size with Systematic Sampling

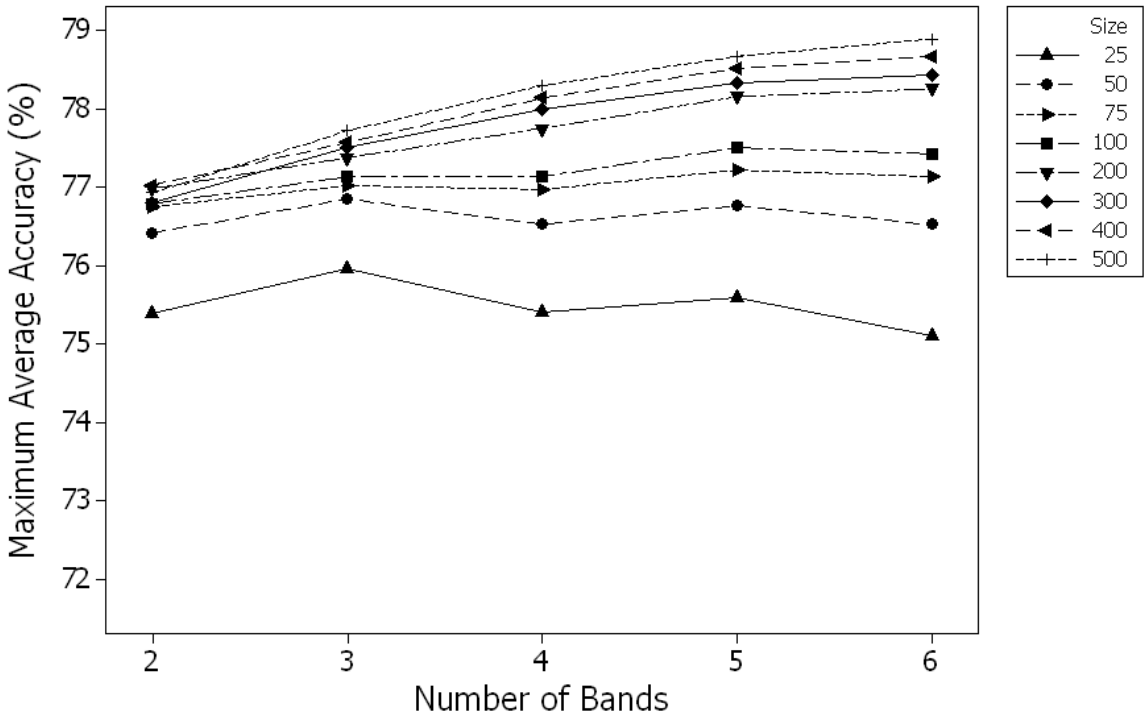


Figure 18d. VA16 Maximum Average Accuracy (%) versus Number of Bands by Sample Size with Random Sampling

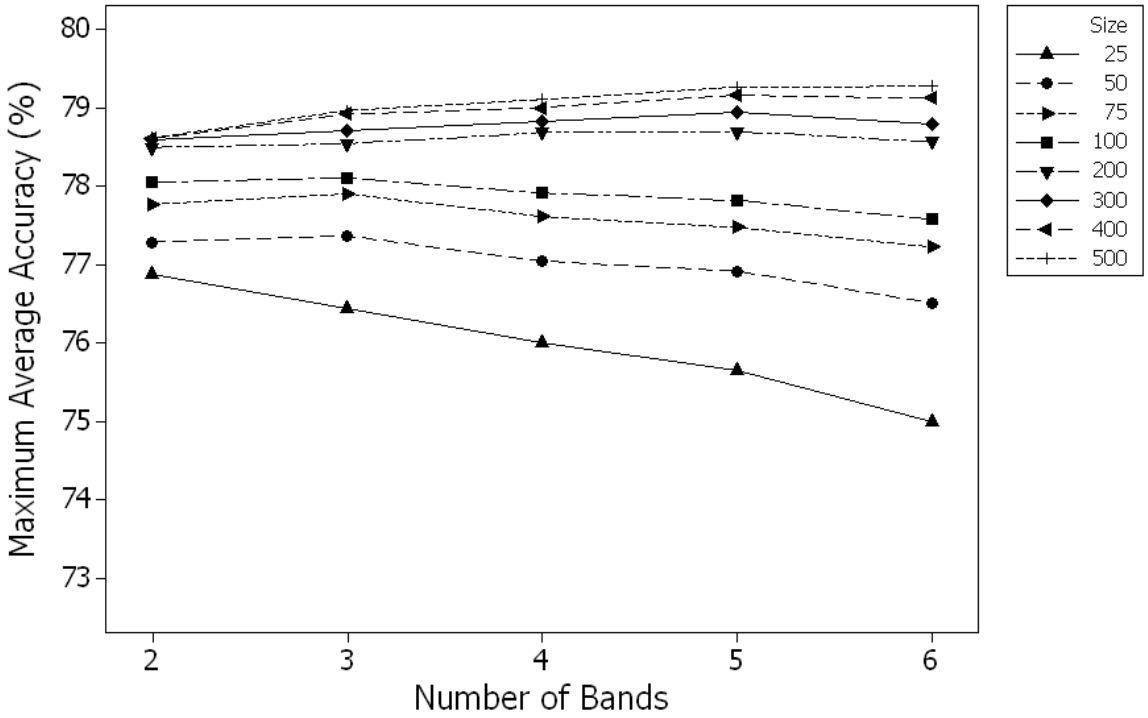


Figure 18e. VA16 Maximum Average Accuracy (%) versus Number of Bands by Sample Size with Stratified Random Sampling

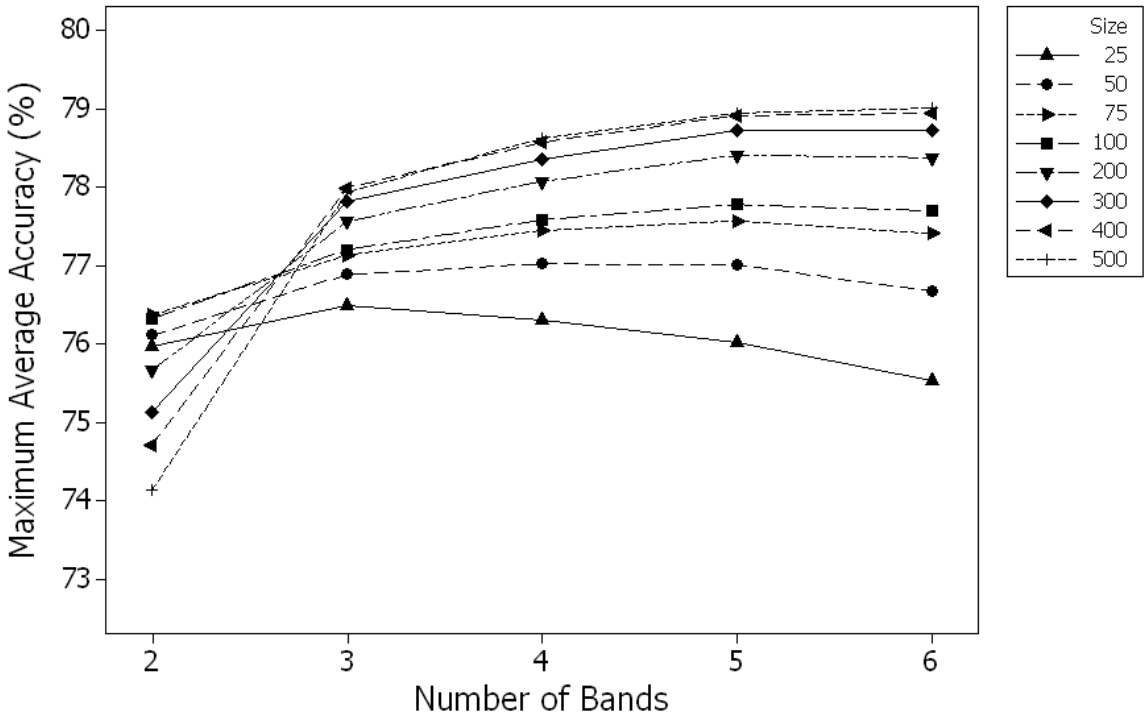


Figure 18f. VA16 Maximum Average Accuracy (%) versus Number of Bands by Sample Size with Systematic Sampling

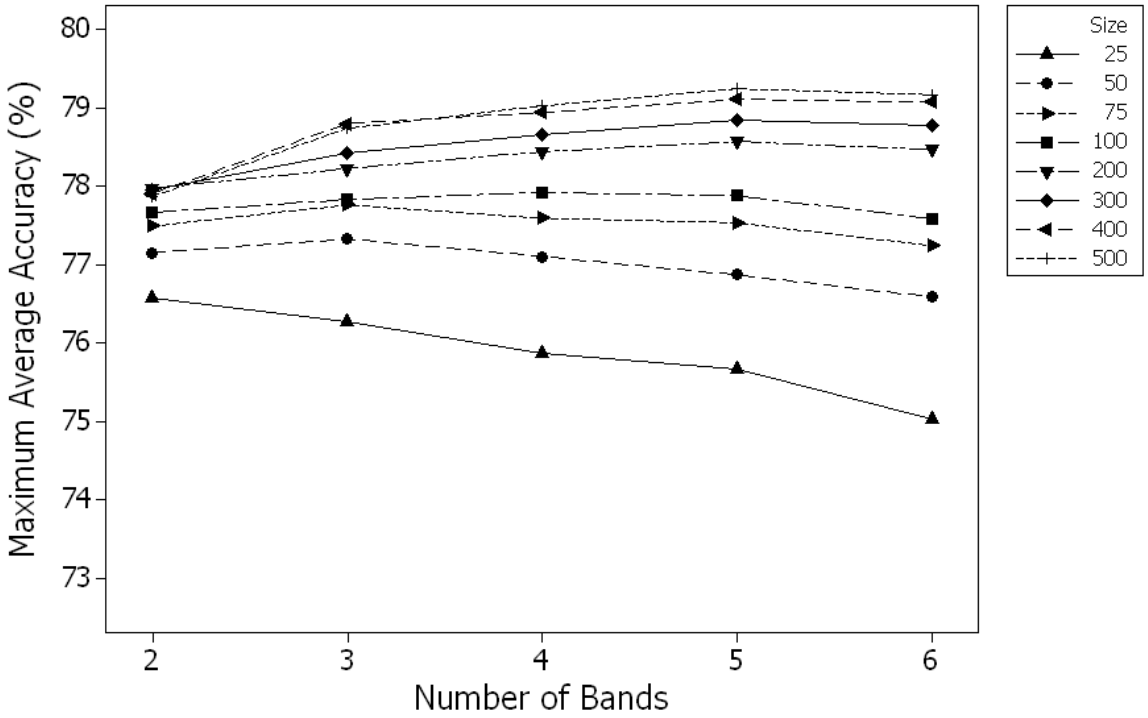


Figure 18g. VA17 Maximum Average Accuracy (%) versus Number of Bands by Sample Size with Random Sampling

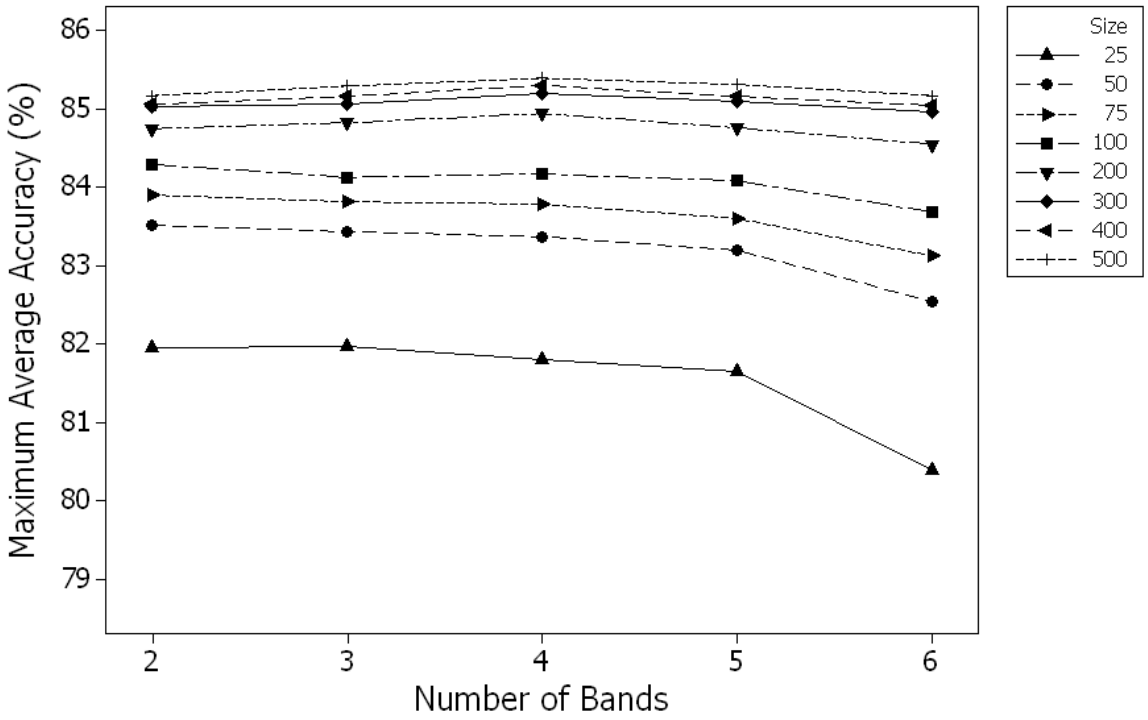


Figure 18h. VA17 Maximum Average Accuracy (%) versus Number of Bands by Sample Size with Stratified Random Sampling

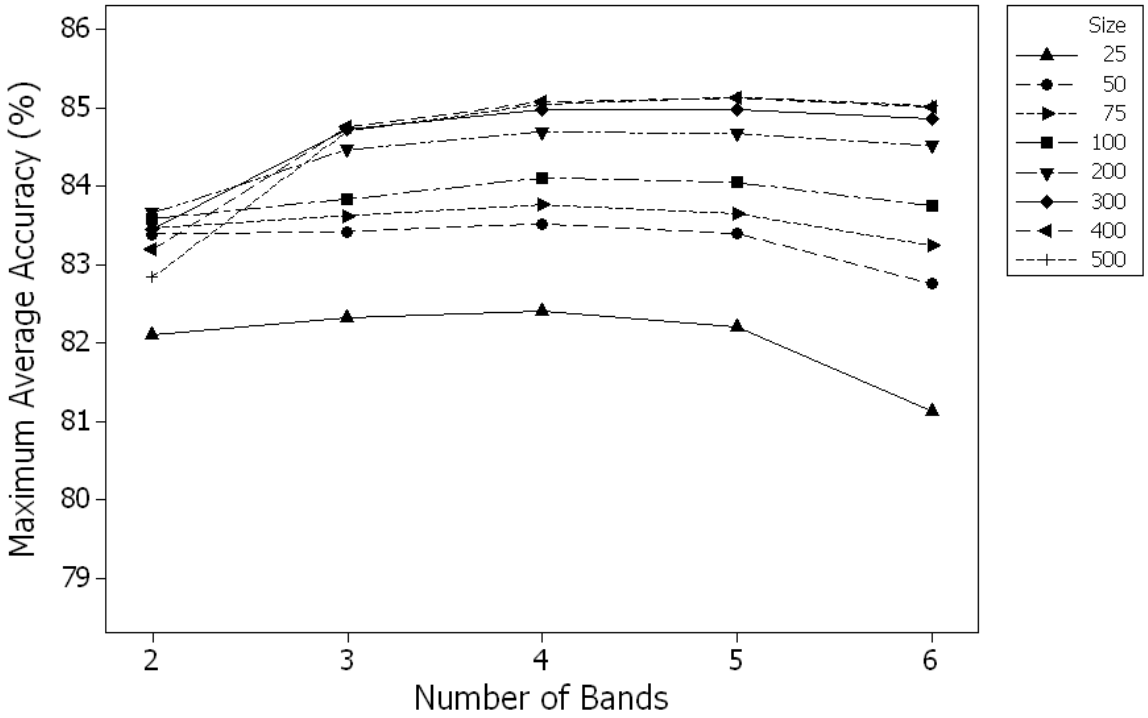


Figure 18i. VA17 Maximum Average Accuracy (%) versus Number of Bands by Sample Size with Systematic Sampling

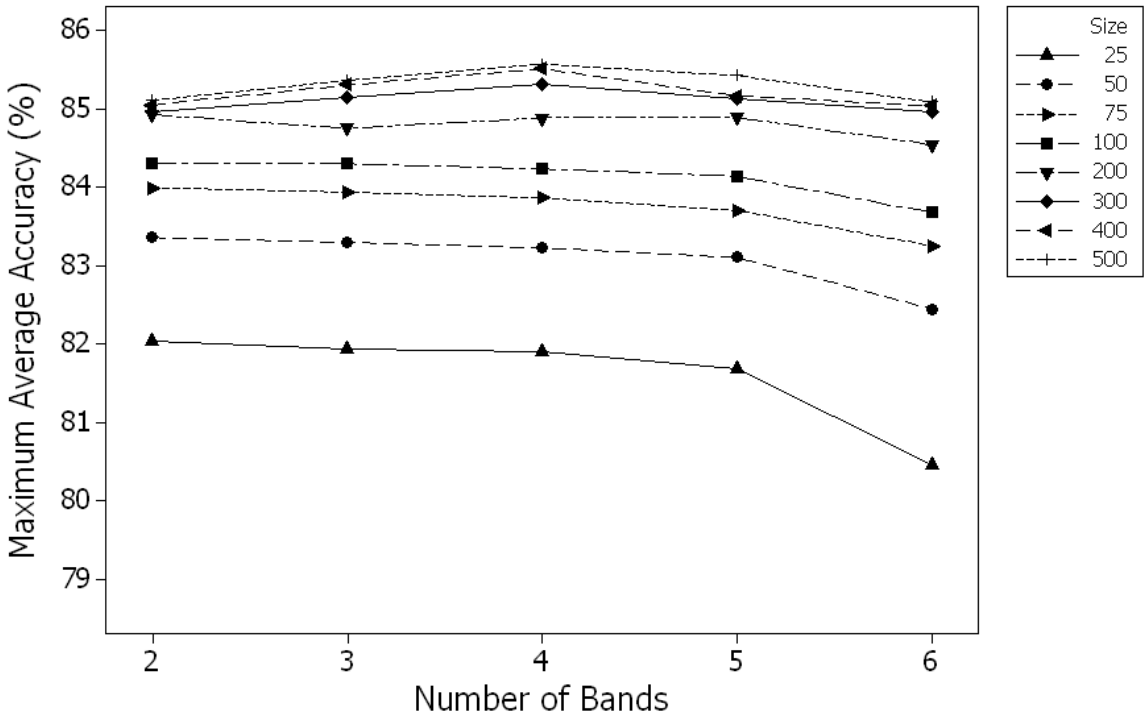


Table 12. Summary of Band Combinations that Resulted in Maximum Average Accuracies for a Given Combination of Image, Sampling Method, and Sample Size

Number of Bands	Band Combination	Proportion of Factor Combinations
1	2	0.7778
1	5	0.1806
1	3	0.0417
2	2 6	0.4861
2	2 4	0.3750
2	2 5	0.1111
2	1 2	0.0278
3	2 4 6	0.3333
3	1 2 4	0.2500
3	1 2 6	0.2361
3	2 4 5	0.0972
3	2 3 4	0.0694
3	1 2 3	0.0139
4	1 2 4 6	0.4306
4	1 2 3 4	0.2361
4	2 3 4 6	0.1806
4	2 3 4 5	0.1111
4	1 2 4 5	0.0417
5	1 2 3 4 6	0.7639
5	1 2 3 4 5	0.1944
5	2 3 4 5 6	0.0417

Table 13. Correlations with Classification Accuracies by Image and Sample Size for Random Samples

Image	Minimum Euclidian Distance	Average Euclidian Distance	Cross Validation Accuracy	Sample Size(s)
VA15	0.1367	0.2781	0.4205	All
VA16	0.0634	0.1553	0.4220	All
VA17	0.1180	0.2283	0.5907	All
VA15	0.37333	0.40807	0.83499	500
VA16	0.33223	0.28448	0.83671	500
VA17	0.39678	0.3325	0.91284	500

Figure 19. VA15 Cross-Validation Accuracy (%) versus Classification Accuracy (%) by Sample Size for Random Samples

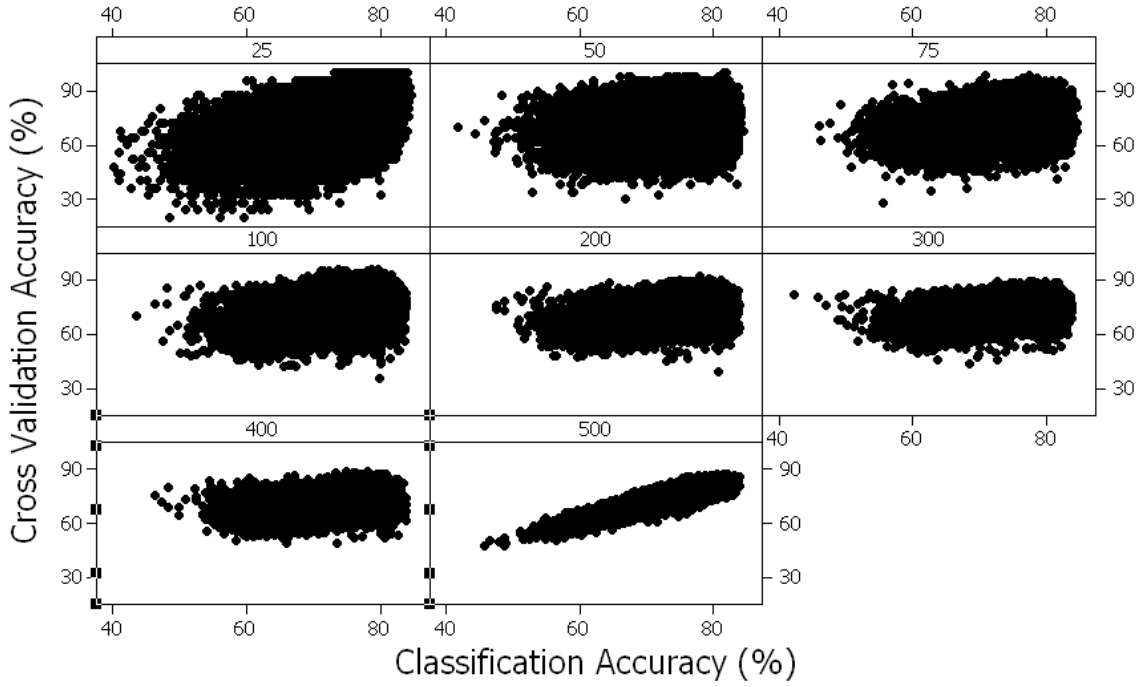


Figure 20. Correlation between Cross-Validation Accuracy and Classification Accuracy versus Sample Size for VA15 with Random Samples

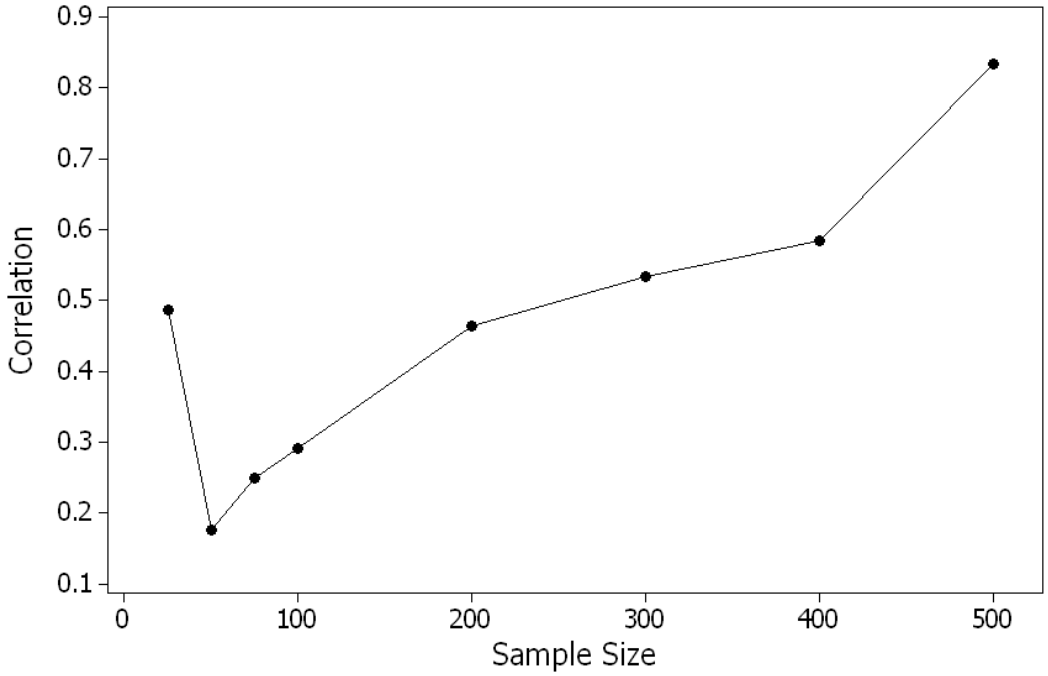
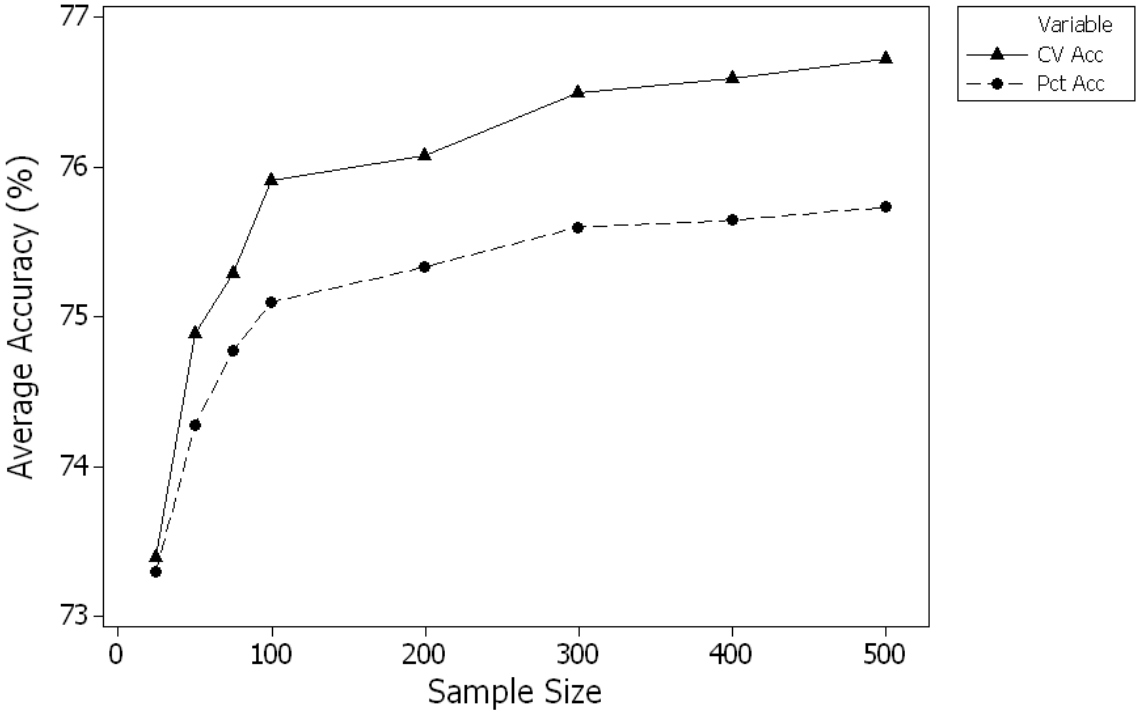


Figure 21. Average Cross-Validation and Classification Accuracy (%) versus Sample Size for VA15 with Random Samples



Variable names: *CV Acc* is the cross-validation accuracy and *Pct Acc* is the classification accuracies.

Chapter 4
DISCUSSION

4.1 Impacts of Sample Size and Sampling Method

4.1.1 Sample Size

The results clearly demonstrated that without additional information to guide the collection of a training data sample a larger training data sample is more likely to result in a higher classification accuracy. This finding is supported by the work of Maselli et al. (1992), Dobbertin and Biging (1996), Arora and Foody (1997), Pal and Mather (2003), and Foody and Mathur (2004a) (Table 1). Unlike the above studies, this work used the nearest neighbor classification technique and a very large number of replicates. Since the variability in classification accuracies declined with sample size increases, the likelihood of producing classifications with low accuracies also decreased (Figures 7a-7c and 8a-8c).

4.1.1.1 Class Proportions

Hardin (1994) used six nonparametric, nearest neighbor techniques and four parametric methods with a full and reduced training data set. In that study, nonparametric techniques had larger declines in accuracy between the full and reduced datasets. This finding is in agreement with larger training data samples performing better with nearest neighbor techniques. The size of the training data sample in Hardin's study was reduced to make the class proportions in the sample differ from the class proportions in the population (i.e., image). When the class proportions matched, nearest neighbor methods outperformed parametric methods (Hardin 1994). Additionally, when the class proportions did not match, the first nearest neighbor classifier, which was used in this study, was one of the superior nonparametric classifiers (Hardin 1994). Since the proportion of forest pixels in the training data samples in this study was more variable with smaller sample sizes, this may partially explain the wider range of

accuracies with smaller samples. Although neither sample size nor the forest proportion in a training sample were highly predictive of classification accuracy, the forest proportion did have a higher correlation with accuracy, 0.484 for forest proportion versus 0.307 for sample size, across all images.

4.1.1.2 Potential of Smaller Sample Sizes

Despite lower average accuracies with smaller training data samples, highly accurate classifications were produced by samples as small as twenty-five pixels (Table 8). Clearly, there is the potential to use smaller training data samples and still get highly accurate results. This finding is not unique to the use of a nearest neighbor classifier. Foody and Mathur (2004b) found similar results in their work with SVMs. Although a soil layer was found useful in identifying the most important training data samples in their study of agricultural fields in an area of two soil types, soil is not a likely candidate in this work. Soil is not seen by the sensor under many of the land covers present in the study area, and is extremely variable across the study area.

What makes a number of small training data sample draws perform significantly better than others still needs to be determined. Without this additional information, the increased likelihood of not producing an accurate classification with a small training data sample will outweigh the effort required to collect a larger sample for most applications.

4.1.1.3 Sample Size Comparisons Across Images

Since the area covered byVA17 was approximately 64% and 51% of the area covered by VA15 and VA16 (Table 3), respectively, it may not be appropriate to compare the impact of sample size across images without an adjustment for image area. Although the size of VA17 may be a factor in the higher classification accuracies observed with this image, the less variable

landscape and the slight increase in the proportion of forest in VA17, are more likely explanations. This is supported by the fact that the average classification accuracies for VA17 with a sample size of twenty-five were higher than those with a sample size of five hundred for both VA15 and VA16 (Figure 9). Since VA17 was acquired in March versus November for the other two images, phenological differences between these two dates and the closer temporal proximity to the VBMP imagery with VA17 may have contributed to the observed differences in classification accuracies.

4.1.2 Sampling Method

Although the sampling methods did not produce major differences in accuracies, the slightly higher average accuracy of all classifications, for a given image, with stratified random sampling may be related to the proportion of forest pixels in a sample draw. The proportion of forest pixels in a stratified random sample was held constant for each sample size, at approximately the proportion of forest in the training data pool. This property gave stratified random samples, with smaller sample sizes, an advantage over samples, of the same size, that were created with the other two sampling methods. As the sample size increased, the variability of the proportion forest in a training sample decreased for the other two sampling methods. With larger sample sizes, random and systematic samples were more likely to result in the highest average accuracy (Figures 6a-6c). The importance of the class proportions in a training sample being similar to the class proportions in the image with nearest neighbor techniques has been shown by Hardin (1994), as discussed in section 4.1.1.1.

Differences in classification accuracies observed between samples drawn with the three sampling methods used in this study were not large enough to warrant the use of one sampling

method over the others. Attention to training sample class proportions is warranted when selecting a sampling method for use with a nearest neighbor classifier.

4.1.3 Other Influences on Classification Accuracy

4.1.3.1 Training Data Pool Issues

A number of factors influenced whether the spectral response within the Landsat ETM+ pixel in which a training data point fell corresponded with the land use interpretation of that training point. These factors included: (1) the accuracy with which both the Landsat ETM+ imagery and the VBMP imagery were registered to a common coordinate system, (2) the distance the point fell from the edge of a land use, (3) the size in terms of dimensions and total area covered by the feature (e.g., road or field) on which the training point was located, and (4) the accuracy of the image interpretation. Since the image interpreter only used the VBMP imagery, which has a 1 meter spatial resolution, to assign a class label to the points in the training data pools, agreement between the class label and the Landsat ETM+ pixel is not guaranteed. Additionally, differences between land cover and land use, and the FIA definitions of land use (Table 2) also contributed to the existence of pixels in the training data pool that were not representative of their class label. For example, a training data point that fell on a narrow row of trees between two fields would be labeled as a nonforest land use even though the pixel contained tree cover.

Since approximately four thousand VBMP images covered the area within the three mosaicked Landsat images, efforts to improve or verify the accuracy with which the image sets were coregistered were not undertaken. Although the VBMP imagery was used as a reference source for ground control points in the rectification and mosaicking of the Landsat imagery, this does not guarantee perfect alignment between the sets of images. Visual inspection of both

training and validation points over both types of images did indicate that registration errors were a contributing factor to some of the observed classification errors. Despite these observations, further adjustments to improve the coregistration of the image sets are unlikely to be made in most operational contexts.

Since the likelihood with which a training point falls on or close to the edge between two land uses increases with the degree of fragmentation in the landscape and the average parcel size, the potential for errors in the training data pool increases with the degree of fragmentation. A high and increasing percentage of non-industrial private landowners of forested tracts within Virginia, coupled with decreases in the average parcel size, has resulted in increased fragmentation, especially in the eastern and northern parts of the state. National forests, including the Washington and Jefferson National Forests, within the western parts of the state, along with lower population densities, have buffered these areas from rapid increases in fragmentation. The observed differences in accuracies between VA17 (from the western part of the state) and VA16 and VA15 (in the central and eastern parts of the state) may also be a result of a larger number of errors in the training data pools for these regions. This suggestion is supported by the smaller proportion of pixels that were edited from the training data pool of VA17 versus the other two images (Table 7).

4.1.3.2 Nearest Neighbor Classifier

In this study, the selection of a classification decision rule was partially based on what was an appropriate choice for use with the available training data set. Since single-pixel training does not result in an estimate of the covariance matrix for a given informational class without additional processing, which would require the subjective choice of parameters for clustering, the simplicity of a nearest neighbor classification rule was deemed most appropriate. The ability to

directly relate a change in classification accuracy with a change in sample size or sampling method without confounding the relationship with other variables was desired.

Since nonparametric methods tend to be more sensitive to the existence of mislabeled pixels in the training data (Cortijo and Perez de la Blanca 1997, Sanchez et al. 2003), the potential of errors in the training data, as discussed in the previous section, may explain why the overall classification accuracies were not as high as expected. Musy et al. (in press) reported overall classification accuracies using the Iterative Guided Spectral Class Rejection (IGSCR) technique, a hybrid classification method, for leaf-off Landsat ETM+ images, corresponding to parts of VA15, VA16, and VA17, of 82.69%, 87.08%, and 91.20%, respectively, using similar validation data. Additional improvements in accuracy resulted from the use of only homogeneous FIA ground plots in the validation stage and post-processing with a 3x3 majority filter (Musy et al., in press). Although the maximum classification accuracies for each image in this study approached those of the Musy et al. study, the average classification accuracies were lower.

Hardin (1994) showed that nearest neighbor classification accuracies with training samples with class proportions similar to the image were higher when more than one nearest neighbor was used. This suggests that additional accuracy improvements with the nearest neighbor classification rule could be realized if more than one neighbor is considered. The utility of the k-nearest neighbor rule for the estimation of forest proportion has already been demonstrated by McRoberts et al. (2002b).

4.1.3.3 Validation Considerations

Some of the other studies that have used FIA plot data for purposes of training or validating an image classification, have treated the four subplots within a single ground plot as

separate data points (McRoberts et al. 2002b). Especially for the purpose of validation, the spatial autocorrelation between the subplots would violate the assumption of independent samples and bias the accuracy calculation if a correction was not made for the autocorrelation. This study only used the land use call at plot center and the total plot proportion forest, which is based on an average of the four subplots. The four subplots, which consisted of a center plot and 3 additional plots, covered an area approximately equivalent to a 3x3 pixel window. Since attributes from the entire plot were used to calculate forest area estimates with the Cochran method, consideration of the class assignments in a 3x3 pixel window around each plot, instead of just the center pixel, may have been warranted.

As with the training data, discrepancies between the land cover in a pixel and the FIA definition of the land use on the ground caused some of the classification errors. Tree cover in residential areas is especially problematic when attempting to estimate land use from a land cover based classification. Although misregistration errors in the imagery and GPS errors in plot locations have been shown to only be slightly detrimental to the precision of forest area estimates (McRoberts et al. 2002a), these errors still impact the validation process. Verbyla and Hammond (1995) have reported conservative bias in classification accuracies based on pixel based comparisons as a result of both positional errors and differences between the pixel size and minimum mapping unit used to obtain reference data.

4.2 Forest Area Estimation

Without knowledge of the true proportion of forest in each image, a conclusion as to which forest area estimation method resulted in more accurate estimates is not possible. As mentioned earlier, both the forest map marginals and the proportion of pixels labeled forest in the training data pools are biased estimates of the proportion of forest in each image. Although the

FIA ground plots have ground verified land use calls, the proportion of forest plots in the validation sets, or the average proportion of forest observed at each ground plot, are based on very small samples and are also not reliable estimates of the actual forest area. Comparisons between these numbers and the resulting forest area estimates are interesting and reassuring, but are not a guarantee of accuracy.

The Cochran method of calculating forest area estimates resulted in smaller precision estimates than the Card method with the same classification and error matrix. As with overall classification accuracy, larger training data sample sizes produced more precise forest area estimates, and sampling method had a negligible impact. Since the forest area estimates did not vary much between different sample sizes, the ability to produce precise forest area estimates with smaller training data sample sizes was dependent on the ability to produce accurate classifications with those sample sizes. The FIA three percent per million acre precision requirement for forest area estimates was only achieved with some of the classifications of VA17 (Figure 14c). Additional improvements in classification accuracies that will result in more precise forest area estimates are expected once the issues described in section 4.1.3 are addressed.

4.2.1 Differences between Card and Cochran Forest Area Estimates and Precisions

The Card forest area estimates were always larger than the Cochran forest area estimates. This can be explained by comparing the equations (2 and 4) used to calculate each estimate and the characteristics of the data used in the equations. Since M_i and W_j are both map marginals from the classification, they can be removed from the equations. So, in order for the area estimates with the two methods to be the same, the following two conditions must hold true:

$$\hat{P}_1 = \frac{n_{21}}{(n_{21} + n_{11})} \quad (9) \quad \text{and} \quad \hat{P}_2 = \frac{n_{22}}{(n_{12} + n_{22})} \quad (10)$$

Equation 9 means that the average proportion of forest from FIA ground plots that were classified as nonforest would have to equal the proportion of FIA ground plots that were classified as nonforest that really were forest (based on center land use). Equation 10 means that the average proportion of forest from FIA ground plots that were classified as forest would have to equal the proportion of FIA ground plots that were classified as forest that really were forest. The area estimates would also be the same if the differences between the elements on the right and left hand sides of the above equations canceled out when multiplied by the appropriate map marginal and summed. With the data in this study, the right hand side of Eq. 9 was usually greater than the left hand side, and the left hand side of Eq. 10 was almost always greater than the right hand side. Since all three images were more than fifty percent forested, the forest map marginals were almost always larger than the nonforest map marginals in every classification. Under these circumstances, the differences between the values on each side of equations 9 and 10 were unable to cancel each other out. A similar comparison can be made between equations 3 and 5, which are the key components of precision. In this case, both the nonforest and forest variance components are larger with the Card estimation method. The combination of the larger proportion forest estimate and the larger variance of the proportion estimate with the Card method, resulted in the Card estimates having higher estimates of precision than the Cochran estimates (Eq. 7).

4.3 Training Data Editing

Training data pixels in heterogeneous areas were expected to have higher average spectral distances between them and their eight neighbors. Pixels with higher spectral distances frequently did not correspond to pixels on the edge of two or more land uses. Since many land uses in the nonforest class have higher spectral variability than those in the forest class, the

average spectral distance between a training data pixel and its neighbors was not very useful. Forest training data points, with higher average spectral distances, were more likely to be on the edge of forest and nonforest land uses, but a majority of the nonforest training data points with high average spectral distances were in urban areas or bare fields.

Similar results were found with the validation data. The spectral distance between a validation point and its spectral class was not predictive of whether the point was misclassified. Although the average spectral distance of misclassified forest validation points was higher than the average spectral distance of correctly classified forest validation points, significant overlap existed between the distributions of spectral distances for these two outcomes.

Although the above spectral distance measures were unable to identify training data points which were mislabeled or on the edge of two land uses, the misclassification rate frequently identified problem points. The large improvements in the accuracies of classifications created with samples from the edited training data pools confirmed the utility of the editing method (Figures 15a-15c). Despite the effectiveness of this editing technique, the information used to calculate the misclassification rate of a training data point is typically unavailable. Since determination of whether training data editing would result in improved classification accuracies was the goal of this effort, the repeatability of the technique used was not initially considered. In the literature, other training data editing techniques for use with the nearest neighbor classifier have resulted in accuracy improvements (Sanchez et al. 2003), and appear to be viable alternatives to the method used here.

4.4 Band Combinations

4.4.1 All Six Bands versus Band Subsets

Although all six bands were best on average for two of the three images, the per draw results clearly demonstrated that classification accuracy with all six bands was typically inferior to the maximum accuracy with a band subset. The high correlations between bands in the same spectral regions, which are discussed in the next section, were the major reason for this finding. Since the spectral characteristics of a given training data sample draw, even with sample sizes of 500, may have considerable variations from draw to draw, almost any of the different band combinations can result in the highest classification accuracy. This is illustrated by the small percentage of times any given band combination produced the most accurate classification on a per sample draw basis. As Spanner et al. (1984) pointed out, the optimal bands were dependent upon both the image and application. In addition, optimal bands were also dependent on the specific characteristics of a given training data sample draw. One could argue that the sample sizes used in this study were not large enough; however, the decrease in the size of classification accuracy improvements, as the training data sample size increased, suggested otherwise.

4.4.2 Top Band Combinations by Subset Size

In this study, the best single band for discriminating forest from nonforest with all three images was band 2. This finding is contrary to what McRoberts et al. (2002b) found as the single best band for predicting the proportion of forest cover in a pixel with the k-nearest neighbor method and multitemporal imagery for two study areas in Minnesota. The study area that was most heavily forested found band 4, from a November image, to be the best single predictor over all other bands in both the November and May images of the area. A second study area, which was a little more than twenty percent forested, found band 6 from a July image

to outperform the rest of the bands in the July, October, and March images (McRoberts et al. 2002b). Potential reasons why the visible bands tended to perform better than the infrared bands for single band classifications in this study are discussed in more detail in the image characteristics section below.

The two top band combinations with two bands, included the best single band, 2, and either a near-infrared band, 4, with VA15 or a mid-infrared band, 6, with VA16 and VA17. Band combination 2,4 with the November image was also the best two band combination for the more heavily forested area in the McRoberts et al. (2002b) study. In addition to the known utility of near-infrared band 4, the fact that band 4 had the smallest correlation of the other five bands with visible band 2, further explains its selection (Table 14). Increased separation between agricultural land cover and forest is evident in the grey scale image of band 4 in Figure 22a (Figure 22b shows band 4,3,2 composite for comparison). Similarly, mid-infrared band 5 with VA16 and VA17 had correlations with band 2 that were not the smallest, but similar in comparison to correlations with bands 4 and 6. As with band 4, band 5 is frequently cited as an important band in vegetation studies, and also resulted in good separation of natural versus man-made features using leaf-on Landsat TM data in the Washington, DC area by Ormsby (1992).

Highest average accuracies of three band combinations were with 2,4,6 for both VA15 and VA17, and with 2,3,4 for VA16. Combination 2,4,5, which has been recommended along with 3,4,5 by Benson and DeGloria (1985) for photo-interpretation of forest types with leaf-on imagery, also had average accuracies very close to the above mentioned combinations. Since bands 3 and 4 are often selected for vegetation studies, and are also used to create vegetation

Table 14. Correlations between Bands in Each Image

Image	Band	1	2	3	4	5	6
VA15	1	1.0000	0.9481	0.8933	0.2968	0.6519	0.7810
VA15	2	0.9481	1.0000	0.9365	0.4167	0.7042	0.8137
VA15	3	0.8933	0.9365	1.0000	0.4779	0.8268	0.8944
VA15	4	0.2968	0.4167	0.4779	1.0000	0.6402	0.5103
VA15	5	0.6519	0.7042	0.8268	0.6402	1.0000	0.9481
VA15	6	0.7810	0.8137	0.8944	0.5103	0.9481	1.0000
VA16	1	1.0000	0.9323	0.8968	0.6371	0.7318	0.7897
VA16	2	0.9323	1.0000	0.9464	0.7539	0.7935	0.8345
VA16	3	0.8968	0.9464	1.0000	0.7319	0.8710	0.8998
VA16	4	0.6371	0.7539	0.7319	1.0000	0.7517	0.6730
VA16	5	0.7318	0.7935	0.8710	0.7517	1.0000	0.9639
VA16	6	0.7897	0.8345	0.8998	0.6730	0.9639	1.0000
VA17	1	1.0000	0.9472	0.9013	0.6466	0.6985	0.7147
VA17	2	0.9472	1.0000	0.9338	0.7601	0.7425	0.7358
VA17	3	0.9013	0.9338	1.0000	0.6825	0.8564	0.8646
VA17	4	0.6466	0.7601	0.6825	1.0000	0.6734	0.5752
VA17	5	0.6985	0.7425	0.8564	0.6734	1.0000	0.9727
VA17	6	0.7147	0.7358	0.8646	0.5752	0.9727	1.0000

indices, their utility in the discrimination between forest and nonforest is not unexpected. The combination of 2,4,6 has the desirable property of including one band from each of the spectral regions, visible, near-infrared, and mid-infrared. Spanner et al. (1984), in their study of leaf-on thematic mapper simulator data, also found one band from each of the spectral regions, including thermal, to be optimal for investigating forest structure.

Bands 2 and 4 occurred in each of the top combinations with four bands, which were 2,3,4,6, 1,2,4,5, and 1,2,4,6 for VA15, VA16, and VA17, respectively. With band combinations of four or five bands, when band 5 was included, the same combination with band 6 usually produced the next highest average accuracy and vice versa. As with the three band

combinations, at least one band from each of the spectral regions was included in the top four band combinations.

Figure 22a. Band 4 Grey Scale Image of an Agricultural and Forested Area

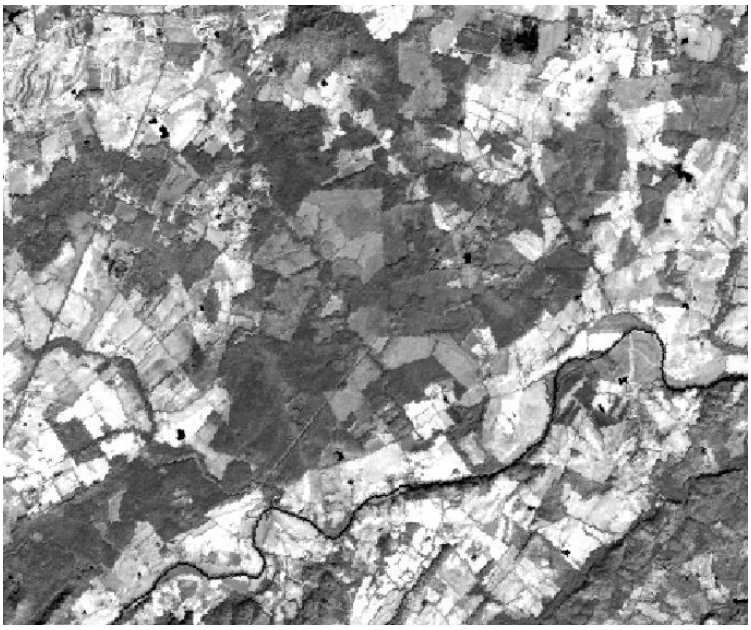
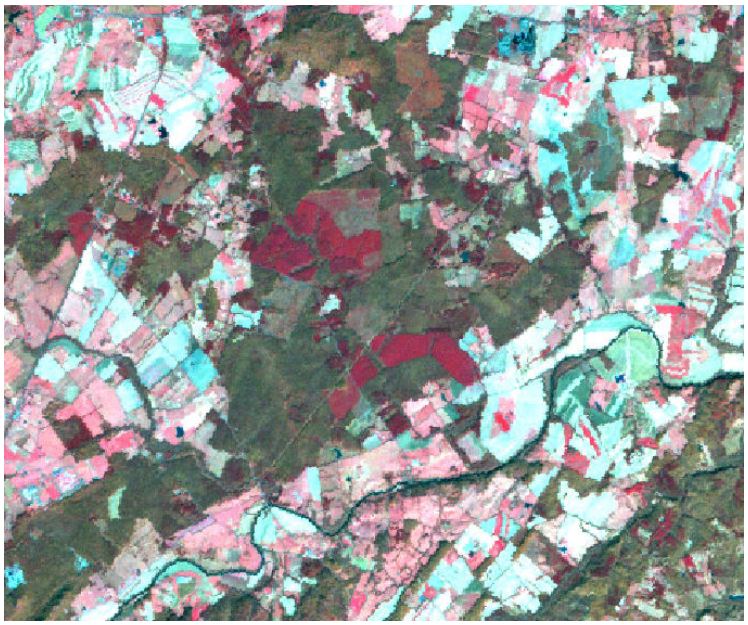


Figure 22b. Band 4,3,2 Composite Image of an Agricultural and Forested Area



Since only one band is not used with five band combinations, the five bands which produce the highest average accuracies will leave out the band that has the least information to add. In this study, combinations 1,2,3,4,5 with VA15 and 1,2,3,4,6 with VA16 and VA17, resulted in the highest average accuracies of a five band combination. Bands 5 and 6 were highly correlated with each other, with correlation values that were higher than all other pairs of bands with VA16 and VA17. The similarity in correlations between band pairs 5,6 and 1,2 with VA15, explains why combination 2,3,4,5,6 was the second top five band combination for this image.

4.4.3 Influence of Image Characteristics

Since high accuracies with VA17 were achievable with much smaller training data sample sizes and fewer bands, there appears to be a decrease in the spectral variability of the informational classes within this image. Although this seems contrary to the finding that four of the six bands in VA17 have larger variances than the same bands in the other two images (Table 15), this may be a result of greater spectral differences between the two informational classes, which may be an indication of fewer mixed pixels. Since mixed pixels typically occur along the edge of land uses and were likely to be edited out of the training data pools, the decrease in the percentage of pixels removed from the training data pool for VA17 supports this deduction.

The prevalence of band 2 in all of the band combinations which produced the highest average classification accuracies may be related to the time of year the imagery was obtained. Since all three images were acquired prior to spring green-up, the deciduous forests were leaf-off and many of the agricultural fields were fallow or contained less vegetative cover than would be present at other times of the year. This reduction in vegetation, which is often most easily discriminated with infrared bands, may explain the greater importance of visible band 2. Both

Table 15. Variance-Covariance Matrix for Each Image

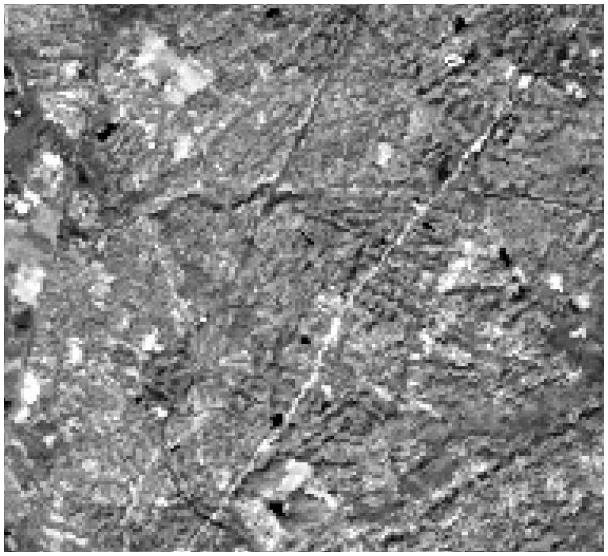
Image	Band	1	2	3	4	5	6
VA15	1	42.899	53.676	81.559	33.018	107.432	93.775
VA15	2	53.676	74.707	112.837	61.162	153.161	128.928
VA15	3	81.559	112.837	194.306	113.136	290.009	228.569
VA15	4	33.018	61.162	113.136	288.433	273.589	158.866
VA15	5	107.432	153.161	290.009	273.589	633.167	437.364
VA15	6	93.775	128.928	228.569	158.866	437.364	336.079
VA16	1	27.357	36.813	59.375	54.633	87.756	60.673
VA16	2	36.813	56.993	90.440	93.318	137.344	92.540
VA16	3	59.375	90.440	160.242	151.907	252.803	167.319
VA16	4	54.633	93.318	151.907	268.810	282.585	162.094
VA16	5	87.756	137.344	252.803	282.585	525.680	324.652
VA16	6	60.673	92.540	167.319	162.094	324.652	215.788
VA17	1	51.605	64.989	92.308	58.876	136.159	93.812
VA17	2	64.989	91.219	127.145	92.023	192.446	128.419
VA17	3	92.308	127.145	203.237	123.331	331.295	225.223
VA17	4	58.876	92.023	123.331	160.673	231.613	133.225
VA17	5	136.159	192.446	331.295	231.613	736.364	482.330
VA17	6	93.812	128.419	225.223	133.225	482.330	333.911

urban and agricultural areas generally had higher band 2 reflectance values than the leaf-off deciduous and conifer forest stands, while water had lower band 2 reflectance than all other land covers. Based on a visual comparison of different land use classes with grey scale images of one band at a time, there clearly was reduced confusion between urban areas with tree cover and forest with band 2 versus band 4 (Figures 23a-23b).

Although leaf-off imagery has been shown to perform well in Virginia for the discrimination of forest and nonforest (Scrivani et al. 2000, Musy et al. in press), high classification accuracies have also been obtained with imagery from May and September (Wayman et al. 2001). Since numerous studies have demonstrated the ability of multitemporal image sets to increase classification accuracies for many applications, including forest area

Figure 23a. Band 2 Grey Scale Image of Urbanized Area with Tree Cover

Figure 23b. Band 4 Grey Scale Image of Urbanized Area with Tree Cover



estimation (e.g., Rack 2000, McRoberts et al. 2002b), when feasible, the use of multitemporal imagery could be a better option. Since Musy et al. (in press) produced forest/nonforest maps with accuracies that were slightly higher than the maximum accuracies observed in this study with imagery from similar dates, the quality and size of the training data samples or the classification algorithm may be more important factors. The classification accuracy differences may be related to the fact that the images used in the Musy et al. (in press) study did not contain

the portion of Virginia in Paths 15 and 16, Row 33 (Figure 1), which correspond to the highly populated Northern Virginia area. Confusion between the land cover of trees in many residential areas, which is a nonforest land use, and the land use of forest, was particularly problematic.

Use of only variance as an indication of spectral band information content will seldom result in the best band selection as stated in Mausel et al. (1990). In this study, band 2, which had the second smallest variance of all bands, produced the best classifications with just one band and was included in each of the top band combinations. The utility of separability indices and cross-validation accuracies in the prediction of classification accuracies is examined below.

4.5 Band Selection Methods

The band selection techniques explored in this study did not select a band combination which would result in improved classification accuracy over the use of all six Landsat ETM+ bands for a majority of the classifications. High classification accuracies were observed across the entire range of values for each separability metric, minimum and average Euclidian distances. Samples with larger values of these metrics are typically expected to produce more accurate classifications. With the nearest neighbor classifier, if samples from each class are on opposite sides of the decision boundary, it may be better if the samples are spectrally closer to one another. This paradox may explain the inability of the separability indices to select a band combination that results in higher classification accuracies than all six bands.

Although the cross-validation accuracies were not highly correlated with the classification accuracies across all sample sizes, the correlation between the two accuracies were high for training data samples of size 500. Even though the correlation was high, the band combination selected by the cross-validation accuracy, for samples of size 500, still did not result in better classification accuracies than achievable with all six bands. The potential gain in

accuracy by using a smaller band combination may not be worth the effort to select an appropriate combination for some applications. For the purposes of forest area estimation, any improvement in classification accuracy results in more precise area estimates, and thus, band selection is desirable.

If even seemingly small increases in accuracy, on the order of half a percent or more, are deemed important for a given application, it is recommended that all possible band combinations of a subset size of three or larger be evaluated. With Landsat ETM+ data, today's computer power is more than adequate to compute all possible band combinations. Since 500 was the largest sample size explored in this study, the use of an even larger training data sample size with cross-validation may prove more useful for the selection of a band combination.

Chapter 5
CONCLUSIONS AND RECOMMENDATIONS

Given the following study parameters:

- Nearest neighbor classifier
- Forest/Nonforest with FIA definitions
- Point (pixel) samples rather than regions
- Leaf-off imagery

The findings from this study included:

- Larger training data samples, which produced classifications with the highest average accuracies and lowest variance, should be used in the absence of additional information.
- Small training data samples are capable of producing accurate classifications, but are more likely to result in lower classification accuracies.
- Random, stratified random, and systematic sampling are all acceptable methods for training data collection.
- The informational class proportions in a training data sample should be considered when using nearest neighbor techniques, especially with smaller sample sizes.
- Landscape characteristics within an image impact classification accuracies.
- For the imagery used in this study, the Cochran method of stratified analysis with plot proportion forest produced lower forest area precision estimates than the Card method of adjusting map marginals with plot center land use.
- Satellite image classifications are a useful tool to improve the relative efficiencies of forest proportion estimates over the use of FIA ground plots alone.

- Training data sets should be edited to remove mislabeled or mixed class samples, especially when used with the nearest neighbor classification technique.
- All six non-thermal Landsat ETM+ bands are usually not needed to maximize the classification accuracy with a single training data sample.
- The appropriate band subset size depended on the image, training data sampling method, and sample size. Fewer bands were needed with smaller sample sizes, but better accuracies resulted with larger sample sizes.
- Bands 2 and 4 were important components of band combinations, which resulted in the highest average classification accuracies, for the discrimination of forest and nonforest classes with leaf-off imagery.
- Similar band combinations resulted in the highest average classification accuracies across all three images.
- Band combinations performed similarly across training data sample sizes and sampling methods.
- Band covariance and correlation matrices provide insight into the best combination of bands.
- Minimum and average Euclidian distances between informational classes were not useful for the selection of band combinations for classifications with the nearest neighbor decision rule.
- Cross-validation accuracies were strongly correlated with classification accuracies when a sample size of 500 was used, but typically did not select band combinations that outperformed the full band set.

Until additional training data sample characteristics that influence the ability of a sample to produce accurate classifications are identified, larger training data samples should be used with the nearest neighbor classification method. Cross-validation accuracy can be used as an indicator of the potential utility of training data sample of a sufficiently large sample size. Additional work is needed to determine which characteristics of small training data samples result in significantly higher classification accuracies than others.

Band selection is warranted, but appropriate selection methods for use with the nearest neighbor classifier require further investigation. Cross-validation techniques with sample sizes larger than 500 may be useful for band selection purposes. Since the individual characteristics of a training data sample determine the most appropriate band combination, a single set of 'good' combinations cannot be prescribed.

The following recommendations should be explored to improve the accuracy of forest/nonforest classifications with nearest neighbor classifiers: (1) identification of good training data editing techniques, (2) increased automation of training data collection, (3) the use of more than one nearest neighbor, (4) multitemporal imagery, (5) other band selection methods (e.g., stepwise selection), and (6) band weighting.

Chapter 6
LITERATURE CITED

Aha, D.W. and Bankert, R. L. (1996) A comparative evaluation of sequential feature selection algorithms. In Fisher, D. and Lenz, J. H. (eds.), *Artificial Intelligence and Statistics V*, New York: Springer-Verlag.

Arai, K., 1992. A supervised thematic mapper classification with a purification of training samples. *International Journal of Remote Sensing*, 13(11):2039-2049.

Arora, M.K. and G.M. Foody, 1997. Log-linear modelling for the evaluation of the variables affecting the accuracy of probabilistic, fuzzy and neural network classifications. *International Journal of Remote Sensing*, 18(4):785-798.

Barandela, R. and E. Gasca, 2000. Decontamination of training samples for supervised pattern recognition methods. *Lecture Notes in Computer Science: Advances in Pattern Recognition*, 1876:621-630.

Barandela, R. and M. Juarez, 2002. Supervised classification of remotely sensed data with ongoing learning capability. *International Journal of Remote Sensing*, 23(22):4965-4970.

Bauer, M.E., 1976. Technological basis and applications of remote sensing of the earth's resources. *IEEE Transactions on Geoscience Electronics*, 14(1):3-9.

Bauer, M.E., M.M. Hixon, B.J. Davis, and J.B. Etheridge, 1978. Area estimation of crops by digital analysis of Landsat data. *Photogrammetric Engineering & Remote Sensing*, 44:1033-1043.

Beauchemin, M. and K.B. Fung, 2001. On statistical band selection for image visualization. *Photogrammetric Engineering & Remote Sensing*, 67(5):571-574.

Benson, A.S. and S.D. DeGloria, 1985. Interpretation of Landsat-4 thematic mapper and multispectral scanner data for forest surveys. *Photogrammetric Engineering and Remote Sensing*, 51(9):1281-1289.

Campbell, J.B., 1981. Spatial correlation effects upon accuracy of supervised classification of land cover. *Photogrammetric Engineering & Remote Sensing*, 47(3):355-363.

Campbell, J.B., 2002. *Introduction to Remote Sensing*, 3rd edition. Guilford Press: New York.

Card, D.H., 1982. Using known map category marginal frequencies to improve estimates of thematic map accuracy. *Photogrammetric Engineering & Remote Sensing*, 48(3):431-439.

Chavez, P.S., G.L. Berlin, and L.B. Sowers, 1982. Statistical method for selecting Landsat MSS ratios. *Journal of Applied Photographic Engineering*, 8(1):23-30.

Chen, D. and D. Stow, 2002. The effect of training strategies on supervised classification at different spatial resolutions. *Photogrammetric Engineering & Remote Sensing*, 68(11):1155-1161.

Cochran, W.G., 1977. *Sampling Techniques* (3rd edition). Wiley: New York.

Conese, C., G. Maracchi, and F. Maselli, 1993. Improvement in maximum likelihood classification performance on highly rugged terrain using principal components analysis. *International Journal of Remote Sensing*, 14(7):1371-1382.

Congalton, R.G., 1988. Using spatial autocorrelation analysis to explore the errors in maps generated from remotely sensed data. *Photogrammetric Engineering & Remote Sensing*, 54(5):587-592.

Cortijo, F.J. and N. Perez De La Blanca, 1997. A comparative study of some non-parametric spectral classifiers. applications to problems with high-overlapping training sets. *International Journal of Remote Sensing*, 18(6):1259-1275.

Craig, R.G., 1979. Autocorrelation in Landsat data. *Proceedings of the Thirteenth International Symposium on Remote Sensing of the Environment*, Ann Arbor, Michigan, 3:1517-1524.

Curran, P.J., 1988. The semivariogram in remote sensing: an introduction. *Remote Sensing of Environment*, 24:493-507.

Czaplewski, R.L., 1999. Multistage remote sensing toward an annual national inventory. *Journal of Forestry*, 97(12):44-48.

Czaplewski, R.L. and G.P. Catts, 1992. Calibration of remotely sensed proportion or area estimates for misclassification error. *Remote Sensing of Environment*, 39:29-43.

Dasarathy, B.V., J.S. Sanchez, and S. Townsend, 2000. Nearest neighbour editing and condensing tools-synergy exploitation. *Pattern Analysis and Applications*, 3(1):19-30.

Deppe, F., 1998. Forest area estimation using sample surveys and Landsat MSS and TM data. *Photogrammetric Engineering & Remote Sensing*, 64(4):285-292.

Dobbertin, M. and G.S. Biging, 1996. A simulation study of the effect of scene autocorrelation, training sample size and sampling method on classification accuracy. *Canadian Journal of Remote Sensing*, 22(4):360-367.

Dymond, J.R., 1992. How accurately do image classifiers estimate area. *International Journal of Remote Sensing*, 13:1735-1742.

Fleming, G.P., P.P. Coulling, K.D. Patterson, and K.M. McCoy, 2004. The natural communities of Virginia: classification of ecological community groups. Second approximation. Virginia Department of Conservation and Recreation, Division of Natural Heritage, Richmond, VA. <http://www.dcr.virginia.gov/dnh/ncintro.htm>

Foody, G.M., 2002. Status of land cover classification accuracy assessment. *Remote Sensing of Environment*, 80(1):185-201.

Foody, G.M. and A. Mathur, 2004a. A relative evaluation of multiclass image classification by support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, 42(6):1335-1343.

Foody, G.M. and A. Mathur, 2004b. Toward intelligent training of supervised image classifications: directing training data acquisition for SVM classification. *Remote Sensing of Environment*, 93:107-117.

Franco-Lopez, H., A.R. Ek, and M.E. Bauer, 2001. Estimation and mapping of forest stand density, volume, and cover type using the k-nearest neighbors method. *Remote Sensing of Environment*, 77:251-274.

Friedl, M.A. and C.E. Brodley, 1997. Decision tree classification of land cover from remotely sensed data. *Remote Sensing of Environment*, 61:399-409.

Gallego, F.J., 2004. Remote sensing and land cover area estimation. *International Journal of Remote Sensing*, 25(15):3019-3047.

Gong, P. and P.J. Howarth, 1990. An assessment of some factors influencing multi-spectral land-cover classification. *Photogrammetric Engineering & Remote Sensing*, 56(5):597-603.

Griffith, J.A., S.V. Stehman, and T.R. Loveland, 2003. Landscape trends in Mid-Atlantic and southeastern United States ecoregions. *Environmental Management*, 32(5): 572-588.

Hardin, P.J., 1994. Parametric and nearest-neighbor methods for hybrid classification: a comparison of pixel assignment accuracy. *Photogrammetric Engineering and Remote Sensing*, 60(12):1439-1448.

Hoppus, M., S. Arner, and A. Lister, 2000. Stratifying FIA ground plots using a 3-year old MRLC forest cover map and current TM derived variables selected by "decision tree" classification. In: Proceedings of the Second Annual Forest Inventory and Analysis (FIA) Symposium, Salt Lake City, UT, October 17-18. (pp. 19-24).

Hughes, G.F., 1968. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55-63.

Jensen, J.R., 2005. *Introductory Digital Image Processing: a Remote Sensing Perspective*, 3rd edition. Prentice Hall: New Jersey.

Katila, M. and E. Tomppo, 2001. Selecting estimation parameters for the Finnish multisource national forest inventory. *Remote Sensing of Environment*, 76:16-32.

Labovitz, M.L. and E.J. Masuoka, 1984. The influence of autocorrelation in signature extraction – an example from a geobotanical investigation of Cotter Basin, Montana. *International Journal of Remote Sensing*, 5(2):315-332.

Lillesand, T.M., R.W. Kiefer, and J.W. Chipman, 2004. *Remote Sensing and Image Interpretation*, 5th edition. John Wiley & Sons, Inc.: New Jersey.

Mäkelä, Helena and A. Pekkarinen, 2001. Estimation of timber volume at the sample plot level by means of image segmentation and Landsat TM imagery. *Remote Sensing of Environment*, 77:66-75.

Maselli, F, C. Conese, L. Petkov, and R. Resti, 1992. Inclusion of prior probabilities derived from a nonparametric process into the maximum-likelihood classifier. *Photogrammetric Engineering & Remote Sensing*, 58(2):201-207.

Mather, P.M., 2004. *Computer Processing of Remotely-Sensed Images: An Introduction*, 3rd edition. John Wiley & Sons, Ltd.: England.

Mausel, P.W., W.J. Kramber, and J.K. Lee, 1990. Optimum band selection for supervised classification of multispectral data. *Photogrammetric Engineering & Remote Sensing*, 56(1):55-60.

McGwire, K., M. Friedl and J.E. Estes, 1993. Spatial structure, sampling design and scale in remotely-sensed imagery of a California savanna woodland. *International Journal of Remote Sensing*, 14(11):2137-2164.

McRoberts, R.E. and M.H. Hansen, 1999. Annual forest inventories for the north central region of the United States. *Journal of Agricultural, Biological and Environmental Statistics*, 4(4):361-371.

McRoberts, R.E., D.G. Wendt, M.D. Nelson, and M.H. Hansen, 2002a. Using a land cover classification based on satellite imagery to improve the precision of forest inventory area estimates. *Remote Sensing of Environment* 81(1):36-44.

McRoberts, R.E., M.D. Nelson, and D.G. Wendt, 2002b. Stratified estimation of forest area using satellite imagery, inventory data, and the *k*-nearest neighbors technique. *Remote Sensing of Environment*, 82(2-3):457-468.

Muasher, M.J. and D.A. Landgrebe, 1984. A binary tree feature selection technique for limited training sample size. *Remote Sensing of Environment*, 16:183-194.

Musy, R.F., R.H. Wynne, C.E. Blinn, J.A. Scrivani, and R.E. McRoberts, in press. Automated forest area estimation via iterative guided spectral class rejection. *Photogrammetric Engineering & Remote Sensing*.

Nadler, M. and E.P. Smith, 1993. *Pattern Recognition Engineering*. John Wiley & Sons Inc.: New York.

Nilsson, M., S. Folving, P. Kennedy, J. Puumalainen, G. Chirici, P. Corona, M. Marchetti, H. Olsson, C. Ricotta, A. Ringvall, G. Stahl and E. Tomppo, 2003. Combining remote sensing and field data for deriving unbiased estimates of forest parameters over large regions. In: *Advances in forest inventory for sustainable forest management and biodiversity monitoring*. 19-32.

Ormsby, J.P., 1992. Evaluation of natural and man-made features using Landsat TM data. *International Journal of Remote Sensing*, 13(2):303-318.

Pal, M. and P.M. Mather, 2003. An assessment of the effectiveness of decision tree methods for land cover classification. *Remote Sensing of Environment*, 86(4):554-565.

Rack, J., 2000. Forest/nonforest classification of Landsat TM data for annual inventory phase one stratification. In: Proceedings of the Second Annual Forest Inventory and Analysis (FIA) Symposium, Salt Lake City, UT, October 17-18. (p8-10)

Raicharoen, T. and C. Lursinsap, 2005. A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (POC-NN) algorithm. *Pattern Recognition Letters*, 26(10):1554-1567.

Reams, G.A. and P.C. Van Deusen, 1999. The southern annual forest inventory system. *Journal of Agricultural, Biological, and Environmental Statistics,* 4(3):108-122.

Richards, J.A. and X. Jia, 1999. *Remote sensing digital image analysis: an introduction*, 3rd edition. Springer: New York.

Sanchez, J.S., R. Barandela, A.I. Marques, R. Alejo, and J. Badenas, 2003. Analysis of new techniques to obtain quality training sets. *Pattern Recognition Letters*, 24:1015-1022.

Scrivani, J.A., R.H. Wynne, C.F. Blinn, and R.F. Musy, 2001. Phase I forest area estimation using Landsat TM and iterative guided spectral class rejection: assessment of possible training data protocols. Proceedings of the 2nd Annual Forest Service FIA Symposium, Salt Lake City, Utah, October 17-18, 2000. General Technical Report SRS-47. Asheville, NC: USDA Forest Service Southern Research Station, pp. 11-14.

Sheffield, C., 1985. Selecting band combinations from multispectral data. P*hotogrammetric Engineering & Remote Sensing*, 51(6):681-687.

Shiver, B.D. and B.E. Borders, 1996. *Sampling Techniques for Forest Resource Inventory*. New York: Wiley.

Spanner, M.A., J.A. Brass, and D.L. Patterson, 1984. Feature selection and the information content of thematic mapper simulator data for forest structural assessment. *IEEE Transactions on Geoscience and Remote Sensing*, 22(6):482-489.

Stehman, S.V., 1992. Comparison of systematic and random sampling for estimating the accuracy of maps generated from remotely sensed data. *Photogrammetric Engineering & Remote Sensing*, 58(9):1343-1350.

Stehman, S.V., 1999. Basic probability sampling designs for thematic map accuracy assessment. *International Journal of Remote Sensing*, 20(12):2423-2441.

Stehman, S.V. and Czaplewski, R.L., 1998. Design and analysis for thematic map accuracy assessment: fundamental principles. *Remote Sensing of Environment*, 64:331-344.

Tomppo, E. and M. Halme, 2004. Using coarse scale forest variables as ancillary information and weighting of variables in k-NN estimation: a genetic algorithm approach. *Remote Sensing of Environment*, 92:1-20.

USDA Forest Service (USDA-FS), 1998. Field instructions for southern forest inventory, B, remeasurement of fixed area plots, version #2, May 1998. Southern Research Station.

USDA Forest Service (USDA-FS), 2005. Fact Sheet Series, Forest Inventory and Analysis Sampling and Plot Design. http://www.fia.fs.fed.us/library/fact-sheets/data-collections/Sampling%20and%20Plot%20Design.pdf.

Verbyla, D.L. and T.O. Hammond, 1995. Conservative bias in classification accuracy assessment due to pixel-by-pixel comparison of classified images with reference grids. *International Journal of Remote Sensing*, 16(3):581-587.

Wayman, J.P., R.H. Wynne, J.A. Scrivani, and G.A. Reams, 2001. Landsat TM-based forest area estimation using iterative guided spectral class rejection. *Photogrammetric Engineering & Remote Sensing*, 67(10):1155-1166.

Webster, R., P.J. Curran and J.W. Munden, 1989. Spatial correlation in reflected radiation from the ground and its implications for sampling and mapping by ground-based radiometry. *Remote Sensing of Environment*, 29(1):67-78.

Wynne, R.H., R.G. Oderwald, G.A. Reams, and J.A. Scrivani, 2000. Optical remote sensing for forest area estimation, *Journal of Forestry*, 98(5):31-36.

Yool, S. R., J.L.Star, J.E. Estes, D.B. Botkin, D.W. Eckhardt, and F.W. Davis, 1986. Performance Analysis of Image-Processing Algorithms for Classification of Natural Vegetation in the Mountains of Southern-California. *International Journal of Remote Sensing*, 7(5):683-702.

Yuan, D., 1997. A simulation comparison of three marginal area estimators for image classification. *Photogrammetric Engineering & Remote Sensing*, 63(4):385-392.

APPENDIX A: Summary Statistics

Table A1.1 VA15 Summary Statistics for Classification Accuracies with All Six Bands by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.7518	0.7600	0.4989	0.8364	0.0485	1000
Stratified	25	0.7556	0.7641	0.5695	0.8364	0.0445	1000
Systematic	25	0.7510	0.7583	0.5057	0.8370	0.0488	1879
Random	50	0.7636	0.7675	0.6326	0.8364	0.0344	1000
Stratified	50	0.7687	0.7721	0.6584	0.8404	0.0319	1000
Systematic	50	0.7653	0.7704	0.6510	0.8341	0.0329	941
Random	75	0.7713	0.7727	0.6728	0.8456	0.0272	1000
Stratified	75	0.7711	0.7732	0.6831	0.8370	0.0270	1000
Systematic	75	0.7714	0.7744	0.6739	0.8347	0.0273	653
Random	100	0.7765	0.7778	0.6871	0.8387	0.0241	1000
Stratified	100	0.7751	0.7773	0.6814	0.8427	0.0232	1000
Systematic	100	0.7743	0.7773	0.6745	0.8307	0.0241	472
Random	200	0.7818	0.7824	0.7130	0.8243	0.0171	1000
Stratified	200	0.7815	0.7824	0.7181	0.8318	0.0176	1000
Systematic	200	0.7826	0.7853	0.7233	0.8175	0.0187	337
Random	300	0.7862	0.7870	0.7319	0.8312	0.0149	1000
Stratified	300	0.7838	0.7847	0.7279	0.8335	0.0155	1000
Systematic	300	0.7844	0.7859	0.7468	0.8243	0.0148	259
Random	400	0.7873	0.7882	0.7405	0.8272	0.0138	1000
Stratified	400	0.7853	0.7859	0.7377	0.8347	0.0140	1000
Systematic	400	0.7868	0.7876	0.7514	0.8169	0.0136	220
Random	500	0.7885	0.7893	0.7457	0.8232	0.0126	1000
Stratified	500	0.7871	0.7876	0.7342	0.8255	0.0129	1000
Systematic	500	0.7890	0.7910	0.7555	0.8152	0.0114	496

Note: The maximum average accuracy for each sample size is highlighted as well as the maximum accuracy of all classifications.

Table A1.2 VA16 Summary Statistics for Classification Accuracies with All Six Bands by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.7499	0.7600	0.5525	0.8360	0.0496	1000
Stratified	25	0.7553	0.7625	0.5474	0.8388	0.0475	1000
Systematic	25	0.7503	0.7574	0.5186	0.8401	0.0490	2328
Random	50	0.7652	0.7691	0.6171	0.8388	0.0352	1000
Stratified	50	0.7668	0.7695	0.6292	0.8406	0.0323	1000
Systematic	50	0.7659	0.7709	0.6454	0.8360	0.0345	1189
Random	75	0.7723	0.7751	0.6543	0.8388	0.0281	1000
Stratified	75	0.7742	0.7763	0.6701	0.8350	0.0263	1000
Systematic	75	0.7724	0.7751	0.6190	0.8262	0.0272	826
Random	100	0.7758	0.7779	0.6863	0.8346	0.0253	1000
Stratified	100	0.7770	0.7790	0.6738	0.8322	0.0240	1000
Systematic	100	0.7760	0.7779	0.6668	0.8304	0.0250	657
Random	200	0.7858	0.7867	0.7203	0.8388	0.0172	1000
Stratified	200	0.7838	0.7844	0.7230	0.8285	0.0171	1000
Systematic	200	0.7848	0.7858	0.7212	0.8318	0.0174	466
Random	300	0.7881	0.7886	0.7351	0.8327	0.0146	1000
Stratified	300	0.7874	0.7881	0.7398	0.8234	0.0146	1000
Systematic	300	0.7879	0.7872	0.7430	0.8295	0.0148	469
Random	400	0.7913	0.7923	0.7440	0.8290	0.0132	1000
Stratified	400	0.7895	0.7895	0.7347	0.8322	0.0135	1000
Systematic	400	0.7908	0.7904	0.7542	0.8188	0.0131	321
Random	500	0.7928	0.7937	0.7579	0.8318	0.0120	1000
Stratified	500	0.7902	0.7909	0.7463	0.8257	0.0121	1000
Systematic	500	0.7918	0.7900	0.7588	0.8169	0.0115	292

Note: The maximum average accuracy for each sample size is highlighted as well as the maximum accuracy of all classifications.

Table A1.3 VA17 Summary Statistics for Classification Accuracies with All Six Bands by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.8039	0.8108	0.6270	0.8874	0.0446	1000
Stratified	25	0.8113	0.8189	0.6604	0.8874	0.0429	1000
Systematic	25	0.8045	0.8117	0.5613	0.8847	0.0458	1209
Random	50	0.8255	0.8297	0.6721	0.8901	0.0309	1000
Stratified	50	0.8276	0.8306	0.6955	0.8838	0.0294	1000
Systematic	50	0.8245	0.8270	0.7081	0.8829	0.0299	635
Random	75	0.8313	0.8342	0.7351	0.8865	0.0256	1000
Stratified	75	0.8325	0.8351	0.7405	0.8928	0.0253	1000
Systematic	75	0.8326	0.8360	0.7505	0.8838	0.0255	411
Random	100	0.8369	0.8396	0.7523	0.8910	0.0221	1000
Stratified	100	0.8375	0.8396	0.7613	0.8847	0.0206	1000
Systematic	100	0.8368	0.8374	0.7631	0.8847	0.0222	336
Random	200	0.8456	0.8459	0.7748	0.8838	0.0161	1000
Stratified	200	0.8452	0.8468	0.7901	0.8874	0.0160	1000
Systematic	200	0.8455	0.8468	0.7901	0.8802	0.0171	286
Random	300	0.8497	0.8505	0.8018	0.8865	0.0136	1000
Stratified	300	0.8486	0.8496	0.7928	0.8856	0.0142	1000
Systematic	300	0.8497	0.8514	0.8036	0.8919	0.0148	336
Random	400	0.8505	0.8505	0.8108	0.8865	0.0119	1000
Stratified	400	0.8502	0.8514	0.8018	0.8901	0.0125	1000
Systematic	400	0.8505	0.8523	0.8180	0.8775	0.0131	411
Random	500	0.8518	0.8527	0.8063	0.8811	0.0119	1000
Stratified	500	0.8503	0.8505	0.8081	0.8811	0.0116	1000
Systematic	500	0.8510	0.8500	0.8234	0.8784	0.0098	496

Note: The maximum average accuracy for each sample size is highlighted as well as the maximum accuracy of all classifications.

Table A2.1 VA15 Statistics for Forest Map Marginal Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.6405	0.6408	0.4158	0.7812	0.0786	101
Stratified	25	0.6360	0.6390	0.5047	0.7837	0.0570	101
Systematic	25	0.6251	0.6297	0.4143	0.8160	0.0744	101
Random	50	0.6158	0.6188	0.4958	0.7134	0.0520	101
Stratified	50	0.6212	0.6202	0.5143	0.7099	0.0471	101
Systematic	50	0.6251	0.6267	0.5119	0.7284	0.0531	101
Random	75	0.6174	0.6180	0.5235	0.7445	0.0464	101
Stratified	75	0.6194	0.6179	0.5370	0.6854	0.0328	101
Systematic	75	0.6146	0.6126	0.5047	0.7248	0.0413	101
Random	100	0.6159	0.6170	0.4869	0.6915	0.0337	101
Stratified	100	0.6088	0.6082	0.5420	0.6823	0.0263	101
Systematic	100	0.6109	0.6176	0.5131	0.6881	0.0373	101
Random	200	0.6043	0.6053	0.5421	0.6552	0.0251	101
Stratified	200	0.6098	0.6097	0.5556	0.6777	0.0217	101
Systematic	200	0.6055	0.6070	0.5422	0.6641	0.0262	101
Random	300	0.6002	0.6011	0.5391	0.6444	0.0204	101
Stratified	300	0.6005	0.5995	0.5703	0.6335	0.0149	101
Systematic	300	0.6012	0.6044	0.5459	0.6410	0.0202	101
Random	400	0.6020	0.6012	0.5652	0.6538	0.0179	101
Stratified	400	0.5946	0.5947	0.5514	0.6402	0.0152	101
Systematic	400	0.6017	0.6017	0.5676	0.6386	0.0170	101
Random	500	0.6015	0.6017	0.5564	0.6428	0.0155	101
Stratified	500	0.5960	0.5965	0.5559	0.6224	0.0119	101
Systematic	500	0.6002	0.6010	0.5659	0.6309	0.0150	101

Table A2.2 VA16 Statistics for Forest Map Marginal Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.6552	0.6528	0.4351	0.8529	0.0880	101
Stratified	25	0.6768	0.6803	0.5000	0.7923	0.0624	101
Systematic	25	0.6499	0.6507	0.4098	0.8571	0.0900	101
Random	50	0.6525	0.6481	0.5099	0.7619	0.0525	101
Stratified	50	0.6521	0.6540	0.5498	0.8085	0.0481	101
Systematic	50	0.6418	0.6426	0.4952	0.7558	0.0548	101
Random	75	0.6479	0.6557	0.5460	0.7647	0.0442	101
Stratified	75	0.6465	0.6493	0.5296	0.7376	0.0379	101
Systematic	75	0.6501	0.6561	0.4603	0.7710	0.0478	101
Random	100	0.6438	0.6489	0.5432	0.7206	0.0398	101
Stratified	100	0.6462	0.6475	0.5593	0.7078	0.0316	101
Systematic	100	0.6462	0.6542	0.5501	0.7615	0.0395	101
Random	200	0.6431	0.6434	0.5577	0.7129	0.0299	101
Stratified	200	0.6360	0.6374	0.5872	0.6951	0.0249	101
Systematic	200	0.6430	0.6447	0.5802	0.6932	0.0270	101
Random	300	0.6381	0.6397	0.5732	0.6822	0.0227	101
Stratified	300	0.6372	0.6375	0.5976	0.6801	0.0179	101
Systematic	300	0.6424	0.6418	0.6028	0.6982	0.0192	101
Random	400	0.6391	0.6409	0.5736	0.6805	0.0193	101
Stratified	400	0.6364	0.6349	0.6057	0.6923	0.0146	101
Systematic	400	0.6400	0.6398	0.6025	0.6890	0.0175	101
Random	500	0.6368	0.6374	0.6062	0.6694	0.0142	101
Stratified	500	0.6315	0.6308	0.5908	0.6636	0.0129	101
Systematic	500	0.6344	0.6342	0.6063	0.6759	0.0146	101

Table A2.3 VA17 Statistics for Forest Map Marginal Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.7467	0.7511	0.5465	0.8944	0.0648	101
Stratified	25	0.7356	0.7368	0.6193	0.8359	0.0428	101
Systematic	25	0.7393	0.7318	0.4998	0.9270	0.0731	101
Random	50	0.7281	0.7316	0.5988	0.8504	0.0465	101
Stratified	50	0.7284	0.7287	0.6273	0.8081	0.0346	101
Systematic	50	0.7262	0.7246	0.6195	0.8119	0.0384	101
Random	75	0.7230	0.7267	0.6221	0.8106	0.0368	101
Stratified	75	0.7156	0.7187	0.6130	0.7779	0.0306	101
Systematic	75	0.7223	0.7263	0.6072	0.7927	0.0333	101
Random	100	0.7197	0.7188	0.6587	0.7875	0.0302	101
Stratified	100	0.7183	0.7208	0.6550	0.7625	0.0255	101
Systematic	100	0.7193	0.7201	0.6382	0.7965	0.0316	101
Random	200	0.7139	0.7154	0.6511	0.7709	0.0228	101
Stratified	200	0.7105	0.7106	0.6685	0.7628	0.0180	101
Systematic	200	0.7093	0.7096	0.6519	0.7600	0.0214	101
Random	300	0.7093	0.7107	0.6661	0.7472	0.0181	101
Stratified	300	0.7094	0.7092	0.6784	0.7422	0.0139	101
Systematic	300	0.7073	0.7063	0.6577	0.7413	0.0169	101
Random	400	0.7093	0.7112	0.6770	0.7391	0.0136	101
Stratified	400	0.7070	0.7066	0.6735	0.7336	0.0136	101
Systematic	400	0.7078	0.7078	0.6794	0.7496	0.0137	101
Random	500	0.7043	0.7047	0.6752	0.7341	0.0126	101
Stratified	500	0.7060	0.7055	0.6664	0.7347	0.0121	101
Systematic	500	0.7073	0.7066	0.6819	0.7529	0.0139	101

Table A3.1 VA15 Statistics for Card Forest Area Estimate Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.6224	0.6202	0.6050	0.6458	0.0092	101
Stratified	25	0.6211	0.6189	0.6097	0.6405	0.0079	101
Systematic	25	0.6233	0.6212	0.6091	0.6414	0.0080	101
Random	50	0.6197	0.6190	0.6082	0.6437	0.0061	101
Stratified	50	0.6192	0.6189	0.6074	0.6396	0.0061	101
Systematic	50	0.6203	0.6193	0.6062	0.6391	0.0065	101
Random	75	0.6187	0.6175	0.6070	0.6451	0.0061	101
Stratified	75	0.6193	0.6190	0.6086	0.6359	0.0050	101
Systematic	75	0.6183	0.6180	0.6087	0.6307	0.0046	101
Random	100	0.6184	0.6184	0.6070	0.6328	0.0046	101
Stratified	100	0.6175	0.6177	0.6066	0.6416	0.0050	101
Systematic	100	0.6181	0.6177	0.6077	0.6345	0.0047	101
Random	200	0.6167	0.6168	0.6070	0.6287	0.0041	101
Stratified	200	0.6165	0.6165	0.6069	0.6269	0.0044	101
Systematic	200	0.6168	0.6163	0.6032	0.6267	0.0043	101
Random	300	0.6165	0.6163	0.6060	0.6267	0.0043	101
Stratified	300	0.6168	0.6166	0.6021	0.6290	0.0043	101
Systematic	300	0.6168	0.6162	0.6062	0.6286	0.0043	101
Random	400	0.6163	0.6170	0.6034	0.6251	0.0048	101
Stratified	400	0.6166	0.6168	0.6062	0.6242	0.0041	101
Systematic	400	0.6158	0.6162	0.6037	0.6243	0.0045	101
Random	500	0.6158	0.6157	0.6074	0.6273	0.0042	101
Stratified	500	0.6158	0.6163	0.6069	0.6274	0.0042	101
Systematic	500	0.6158	0.6154	0.6073	0.6274	0.0042	101

Table A3.2 VA16 Statistics for Card Forest Area Estimate Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.6267	0.6266	0.6157	0.6367	0.0042	101
Stratified	25	0.6268	0.6273	0.6161	0.6369	0.0045	101
Systematic	25	0.6264	0.6263	0.6164	0.6348	0.0041	101
Random	50	0.6261	0.6267	0.6149	0.6332	0.0039	101
Stratified	50	0.6255	0.6256	0.6169	0.6356	0.0038	101
Systematic	50	0.6263	0.6264	0.6159	0.6395	0.0041	101
Random	75	0.6255	0.6256	0.6143	0.6355	0.0042	101
Stratified	75	0.6260	0.6257	0.6159	0.6352	0.0040	101
Systematic	75	0.6259	0.6261	0.6177	0.6334	0.0036	101
Random	100	0.6255	0.6255	0.6170	0.6355	0.0036	101
Stratified	100	0.6256	0.6254	0.6188	0.6344	0.0037	101
Systematic	100	0.6254	0.6258	0.6158	0.6357	0.0040	101
Random	200	0.6247	0.6247	0.6132	0.6361	0.0046	101
Stratified	200	0.6252	0.6254	0.6157	0.6342	0.0036	101
Systematic	200	0.6254	0.6255	0.6134	0.6335	0.0039	101
Random	300	0.6247	0.6242	0.6151	0.6400	0.0045	101
Stratified	300	0.6248	0.6246	0.6168	0.6348	0.0041	101
Systematic	300	0.6248	0.6247	0.6145	0.6350	0.0042	101
Random	400	0.6243	0.6245	0.6147	0.6314	0.0036	101
Stratified	400	0.6247	0.6251	0.6131	0.6353	0.0041	101
Systematic	400	0.6244	0.6248	0.6139	0.6326	0.0035	101
Random	500	0.6243	0.6244	0.6145	0.6348	0.0042	101
Stratified	500	0.6249	0.6250	0.6170	0.6339	0.0036	101
Systematic	500	0.6248	0.6243	0.6178	0.6351	0.0036	101

Table A3.3 VA17 Statistics for Card Forest Area Estimate Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.6715	0.6719	0.6582	0.6837	0.0054	101
Stratified	25	0.6720	0.6722	0.6603	0.6855	0.0050	101
Systematic	25	0.6725	0.6723	0.6584	0.6833	0.0047	101
Random	50	0.6724	0.6728	0.6598	0.6855	0.0054	101
Stratified	50	0.6722	0.6726	0.6623	0.6839	0.0046	101
Systematic	50	0.6717	0.6710	0.6621	0.6818	0.0045	101
Random	75	0.6724	0.6729	0.6539	0.6869	0.0057	101
Stratified	75	0.6732	0.6727	0.6622	0.6841	0.0052	101
Systematic	75	0.6716	0.6714	0.6600	0.6832	0.0049	101
Random	100	0.6727	0.6724	0.6595	0.6857	0.0048	101
Stratified	100	0.6721	0.6725	0.6597	0.6838	0.0050	101
Systematic	100	0.6722	0.6728	0.6582	0.6846	0.0044	101
Random	200	0.6727	0.6730	0.6595	0.6855	0.0047	101
Stratified	200	0.6726	0.6725	0.6582	0.6856	0.0047	101
Systematic	200	0.6733	0.6735	0.6581	0.6843	0.0052	101
Random	300	0.6730	0.6735	0.6578	0.6852	0.0055	101
Stratified	300	0.6726	0.6725	0.6627	0.6854	0.0047	101
Systematic	300	0.6726	0.6726	0.6605	0.6867	0.0052	101
Random	400	0.6737	0.6739	0.6558	0.6926	0.0060	101
Stratified	400	0.6732	0.6735	0.6628	0.6836	0.0044	101
Systematic	400	0.6727	0.6730	0.6615	0.6831	0.0053	101
Random	500	0.6727	0.6730	0.6553	0.6868	0.0054	101
Stratified	500	0.6731	0.6733	0.6584	0.6859	0.0051	101
Systematic	500	0.6729	0.6738	0.6563	0.6840	0.0059	101

Table A4.1 VA15 Statistics for Cochran Forest Area Estimate Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.6176	0.6155	0.6010	0.6406	0.0090	101
Stratified	25	0.6164	0.6141	0.6053	0.6352	0.0077	101
Systematic	25	0.6185	0.6164	0.6046	0.6361	0.0079	101
Random	50	0.6150	0.6142	0.6036	0.6383	0.0060	101
Stratified	50	0.6145	0.6142	0.6030	0.6345	0.0059	101
Systematic	50	0.6156	0.6147	0.6017	0.6339	0.0064	101
Random	75	0.6140	0.6130	0.6028	0.6398	0.0060	101
Stratified	75	0.6146	0.6143	0.6039	0.6308	0.0049	101
Systematic	75	0.6137	0.6135	0.6042	0.6257	0.0045	101
Random	100	0.6137	0.6137	0.6026	0.6277	0.0045	101
Stratified	100	0.6129	0.6131	0.6024	0.6363	0.0049	101
Systematic	100	0.6134	0.6129	0.6031	0.6294	0.0045	101
Random	200	0.6120	0.6123	0.6028	0.6237	0.0040	101
Stratified	200	0.6119	0.6119	0.6025	0.6219	0.0043	101
Systematic	200	0.6121	0.6116	0.5988	0.6217	0.0042	101
Random	300	0.6118	0.6117	0.6018	0.6219	0.0041	101
Stratified	300	0.6122	0.6120	0.5982	0.6241	0.0041	101
Systematic	300	0.6121	0.6115	0.6020	0.6236	0.0042	101
Random	400	0.6116	0.6123	0.5989	0.6202	0.0047	101
Stratified	400	0.6120	0.6122	0.6016	0.6194	0.0040	101
Systematic	400	0.6111	0.6116	0.5993	0.6194	0.0045	101
Random	500	0.6112	0.6111	0.6030	0.6224	0.0041	101
Stratified	500	0.6112	0.6116	0.6026	0.6225	0.0041	101
Systematic	500	0.6111	0.6107	0.6027	0.6225	0.0041	101

Table A4.2 VA16 Statistics for Cochran Forest Area Estimate Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.6245	0.6245	0.6140	0.6338	0.0041	101
Stratified	25	0.6247	0.6252	0.6143	0.6342	0.0043	101
Systematic	25	0.6242	0.6242	0.6146	0.6323	0.0040	101
Random	50	0.6240	0.6245	0.6131	0.6309	0.0038	101
Stratified	50	0.6234	0.6235	0.6151	0.6330	0.0036	101
Systematic	50	0.6241	0.6242	0.6143	0.6367	0.0040	101
Random	75	0.6234	0.6235	0.6124	0.6331	0.0040	101
Stratified	75	0.6239	0.6236	0.6141	0.6327	0.0039	101
Systematic	75	0.6238	0.6240	0.6160	0.6311	0.0034	101
Random	100	0.6234	0.6234	0.6152	0.6330	0.0035	101
Stratified	100	0.6235	0.6232	0.6170	0.6321	0.0035	101
Systematic	100	0.6233	0.6237	0.6141	0.6331	0.0039	101
Random	200	0.6226	0.6226	0.6115	0.6335	0.0044	101
Stratified	200	0.6231	0.6233	0.6140	0.6317	0.0035	101
Systematic	200	0.6233	0.6234	0.6117	0.6311	0.0038	101
Random	300	0.6226	0.6221	0.6133	0.6374	0.0043	101
Stratified	300	0.6228	0.6225	0.6151	0.6324	0.0039	101
Systematic	300	0.6227	0.6226	0.6128	0.6325	0.0040	101
Random	400	0.6222	0.6224	0.6131	0.6290	0.0034	101
Stratified	400	0.6226	0.6230	0.6114	0.6328	0.0040	101
Systematic	400	0.6223	0.6227	0.6123	0.6303	0.0033	101
Random	500	0.6222	0.6224	0.6129	0.6322	0.0040	101
Stratified	500	0.6228	0.6229	0.6152	0.6314	0.0035	101
Systematic	500	0.6228	0.6222	0.6160	0.6326	0.0034	101

Table A4.3 VA17 Statistics for Cochran Forest Area Estimate Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.6624	0.6628	0.6493	0.6740	0.0052	101
Stratified	25	0.6629	0.6631	0.6516	0.6760	0.0049	101
Systematic	25	0.6633	0.6631	0.6495	0.6739	0.0045	101
Random	50	0.6633	0.6637	0.6511	0.6760	0.0053	101
Stratified	50	0.6631	0.6635	0.6534	0.6745	0.0045	101
Systematic	50	0.6625	0.6619	0.6533	0.6722	0.0044	101
Random	75	0.6633	0.6637	0.6453	0.6773	0.0055	101
Stratified	75	0.6641	0.6635	0.6533	0.6747	0.0050	101
Systematic	75	0.6625	0.6623	0.6511	0.6737	0.0048	101
Random	100	0.6635	0.6633	0.6506	0.6763	0.0046	101
Stratified	100	0.6630	0.6634	0.6510	0.6742	0.0049	101
Systematic	100	0.6631	0.6636	0.6495	0.6751	0.0043	101
Random	200	0.6636	0.6639	0.6505	0.6758	0.0046	101
Stratified	200	0.6635	0.6633	0.6492	0.6761	0.0046	101
Systematic	200	0.6642	0.6644	0.6495	0.6748	0.0050	101
Random	300	0.6639	0.6643	0.6491	0.6757	0.0053	101
Stratified	300	0.6635	0.6633	0.6539	0.6762	0.0046	101
Systematic	300	0.6635	0.6635	0.6516	0.6771	0.0051	101
Random	400	0.6645	0.6647	0.6472	0.6827	0.0058	101
Stratified	400	0.6641	0.6643	0.6539	0.6742	0.0043	101
Systematic	400	0.6636	0.6638	0.6526	0.6738	0.0051	101
Random	500	0.6636	0.6638	0.6467	0.6771	0.0052	101
Stratified	500	0.6640	0.6642	0.6495	0.6763	0.0050	101
Systematic	500	0.6638	0.6647	0.6476	0.6746	0.0058	101

Table A5.1 VA15 Statistics for Precision of Card Forest Area Estimate Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.0414	0.0414	0.0362	0.0466	0.0023	101
Stratified	25	0.0414	0.0413	0.0365	0.0465	0.0021	101
Systematic	25	0.0414	0.0412	0.0362	0.0466	0.0022	101
Random	50	0.0410	0.0411	0.0362	0.0457	0.0018	101
Stratified	50	0.0408	0.0408	0.0362	0.0452	0.0017	101
Systematic	50	0.0409	0.0409	0.0369	0.0449	0.0017	101
Random	75	0.0407	0.0407	0.0355	0.0450	0.0015	101
Stratified	75	0.0407	0.0408	0.0363	0.0446	0.0015	101
Systematic	75	0.0407	0.0407	0.0366	0.0451	0.0015	101
Random	100	0.0405	0.0406	0.0362	0.0438	0.0013	101
Stratified	100	0.0406	0.0406	0.0359	0.0449	0.0014	101
Systematic	100	0.0406	0.0405	0.0370	0.0446	0.0013	101
Random	200	0.0402	0.0402	0.0373	0.0437	0.0010	101
Stratified	200	0.0403	0.0403	0.0370	0.0435	0.0011	101
Systematic	200	0.0402	0.0401	0.0380	0.0428	0.0011	101
Random	300	0.0399	0.0399	0.0368	0.0425	0.0009	101
Stratified	300	0.0401	0.0400	0.0371	0.0422	0.0009	101
Systematic	300	0.0401	0.0400	0.0375	0.0420	0.0009	101
Random	400	0.0399	0.0400	0.0373	0.0422	0.0009	101
Stratified	400	0.0400	0.0400	0.0366	0.0422	0.0009	101
Systematic	400	0.0400	0.0400	0.0379	0.0416	0.0008	101
Random	500	0.0398	0.0399	0.0378	0.0420	0.0008	101
Stratified	500	0.0399	0.0399	0.0374	0.0428	0.0008	101
Systematic	500	0.0398	0.0398	0.0379	0.0415	0.0008	101

Table A5.2 VA16 Statistics for Precision of Card Forest Area Estimate Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.0413	0.0410	0.0359	0.0472	0.0024	101
Stratified	25	0.0412	0.0409	0.0359	0.0473	0.0025	101
Systematic	25	0.0412	0.0410	0.0356	0.0468	0.0023	101
Random	50	0.0407	0.0406	0.0360	0.0468	0.0020	101
Stratified	50	0.0407	0.0407	0.0357	0.0462	0.0018	101
Systematic	50	0.0406	0.0404	0.0360	0.0451	0.0019	101
Random	75	0.0404	0.0404	0.0359	0.0448	0.0016	101
Stratified	75	0.0403	0.0402	0.0361	0.0450	0.0016	101
Systematic	75	0.0404	0.0404	0.0370	0.0454	0.0016	101
Random	100	0.0402	0.0402	0.0363	0.0442	0.0015	101
Stratified	100	0.0402	0.0402	0.0366	0.0451	0.0015	101
Systematic	100	0.0402	0.0401	0.0368	0.0448	0.0015	101
Random	200	0.0397	0.0397	0.0359	0.0428	0.0011	101
Stratified	200	0.0398	0.0397	0.0367	0.0426	0.0011	101
Systematic	200	0.0397	0.0397	0.0365	0.0429	0.0011	101
Random	300	0.0395	0.0396	0.0365	0.0422	0.0010	101
Stratified	300	0.0396	0.0395	0.0373	0.0421	0.0010	101
Systematic	300	0.0396	0.0395	0.0368	0.0421	0.0010	101
Random	400	0.0394	0.0394	0.0370	0.0417	0.0009	101
Stratified	400	0.0394	0.0394	0.0365	0.0418	0.0009	101
Systematic	400	0.0394	0.0395	0.0375	0.0416	0.0009	101
Random	500	0.0392	0.0392	0.0367	0.0419	0.0009	101
Stratified	500	0.0394	0.0394	0.0372	0.0424	0.0009	101
Systematic	500	0.0393	0.0394	0.0374	0.0411	0.0008	101

Table A5.3 VA17 Statistics for Precision of Card Forest Area Estimate Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.0366	0.0366	0.0297	0.0433	0.0029	101
Stratified	25	0.0362	0.0362	0.0296	0.0434	0.0029	101
Systematic	25	0.0365	0.0364	0.0299	0.0440	0.0030	101
Random	50	0.0352	0.0351	0.0293	0.0424	0.0023	101
Stratified	50	0.0351	0.0351	0.0302	0.0418	0.0023	101
Systematic	50	0.0354	0.0353	0.0301	0.0417	0.0023	101
Random	75	0.0348	0.0348	0.0296	0.0402	0.0020	101
Stratified	75	0.0347	0.0346	0.0291	0.0403	0.0020	101
Systematic	75	0.0348	0.0348	0.0301	0.0397	0.0021	101
Random	100	0.0344	0.0344	0.0293	0.0392	0.0019	101
Stratified	100	0.0344	0.0343	0.0301	0.0398	0.0017	101
Systematic	100	0.0345	0.0345	0.0300	0.0393	0.0018	101
Random	200	0.0337	0.0339	0.0302	0.0380	0.0014	101
Stratified	200	0.0337	0.0337	0.0297	0.0384	0.0014	101
Systematic	200	0.0337	0.0335	0.0303	0.0370	0.0014	101
Random	300	0.0333	0.0333	0.0297	0.0367	0.0012	101
Stratified	300	0.0335	0.0334	0.0300	0.0377	0.0013	101
Systematic	300	0.0334	0.0331	0.0293	0.0366	0.0013	101
Random	400	0.0333	0.0333	0.0299	0.0366	0.0011	101
Stratified	400	0.0333	0.0333	0.0293	0.0364	0.0012	101
Systematic	400	0.0333	0.0333	0.0305	0.0361	0.0011	101
Random	500	0.0332	0.0331	0.0303	0.0363	0.0011	101
Stratified	500	0.0333	0.0334	0.0303	0.0362	0.0011	101
Systematic	500	0.0333	0.0334	0.0308	0.0356	0.0009	101

Table A6.1 VA15 Statistics for Precision of Cochran Forest Area Estimate Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.0382	0.0383	0.0326	0.0436	0.0025	101
Stratified	25	0.0382	0.0382	0.0330	0.0435	0.0023	101
Systematic	25	0.0382	0.0381	0.0325	0.0435	0.0023	101
Random	50	0.0379	0.0380	0.0326	0.0429	0.0019	101
Stratified	50	0.0377	0.0379	0.0327	0.0425	0.0019	101
Systematic	50	0.0378	0.0376	0.0334	0.0417	0.0018	101
Random	75	0.0376	0.0377	0.0320	0.0422	0.0016	101
Stratified	75	0.0376	0.0376	0.0331	0.0416	0.0016	101
Systematic	75	0.0376	0.0375	0.0333	0.0422	0.0016	101
Random	100	0.0373	0.0373	0.0327	0.0407	0.0014	101
Stratified	100	0.0375	0.0376	0.0325	0.0418	0.0014	101
Systematic	100	0.0375	0.0374	0.0336	0.0416	0.0014	101
Random	200	0.0372	0.0371	0.0342	0.0404	0.0011	101
Stratified	200	0.0372	0.0373	0.0337	0.0405	0.0011	101
Systematic	200	0.0371	0.0370	0.0348	0.0398	0.0011	101
Random	300	0.0368	0.0368	0.0334	0.0396	0.0010	101
Stratified	300	0.0370	0.0370	0.0342	0.0392	0.0010	101
Systematic	300	0.0370	0.0369	0.0344	0.0391	0.0009	101
Random	400	0.0368	0.0368	0.0342	0.0389	0.0009	101
Stratified	400	0.0369	0.0369	0.0339	0.0393	0.0009	101
Systematic	400	0.0369	0.0369	0.0347	0.0385	0.0008	101
Random	500	0.0367	0.0367	0.0347	0.0388	0.0008	101
Stratified	500	0.0369	0.0369	0.0343	0.0398	0.0008	101
Systematic	500	0.0367	0.0367	0.0347	0.0387	0.0008	101

Table A6.2 VA16 Statistics for Precision of Cochran Forest Area Estimate Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.0382	0.0380	0.0329	0.0442	0.0024	101
Stratified	25	0.0380	0.0377	0.0327	0.0442	0.0025	101
Systematic	25	0.0381	0.0379	0.0327	0.0437	0.0023	101
Random	50	0.0376	0.0375	0.0327	0.0435	0.0020	101
Stratified	50	0.0376	0.0376	0.0327	0.0432	0.0018	101
Systematic	50	0.0375	0.0373	0.0331	0.0420	0.0019	101
Random	75	0.0373	0.0373	0.0328	0.0415	0.0016	101
Stratified	75	0.0372	0.0370	0.0331	0.0419	0.0016	101
Systematic	75	0.0373	0.0374	0.0339	0.0422	0.0016	101
Random	100	0.0371	0.0370	0.0332	0.0407	0.0014	101
Stratified	100	0.0371	0.0370	0.0334	0.0420	0.0015	101
Systematic	100	0.0371	0.0371	0.0339	0.0416	0.0015	101
Random	200	0.0366	0.0366	0.0329	0.0395	0.0011	101
Stratified	200	0.0366	0.0366	0.0337	0.0394	0.0011	101
Systematic	200	0.0366	0.0366	0.0334	0.0395	0.0011	101
Random	300	0.0364	0.0364	0.0336	0.0389	0.0010	101
Stratified	300	0.0365	0.0364	0.0341	0.0390	0.0010	101
Systematic	300	0.0365	0.0365	0.0337	0.0390	0.0010	101
Random	400	0.0362	0.0363	0.0341	0.0384	0.0009	101
Stratified	400	0.0363	0.0362	0.0334	0.0388	0.0009	101
Systematic	400	0.0363	0.0362	0.0344	0.0387	0.0009	101
Random	500	0.0361	0.0361	0.0335	0.0391	0.0009	101
Stratified	500	0.0362	0.0363	0.0342	0.0392	0.0008	101
Systematic	500	0.0362	0.0363	0.0344	0.0380	0.0008	101

Table A6.3 VA17 Statistics for Precision of Cochran Forest Area Estimate Proportions by Sampling Method and Sample Size

Method	Sample Size	Mean	Median	Minimum	Maximum	Standard Deviation	Number of Classifications
Random	25	0.0342	0.0342	0.0270	0.0410	0.0030	101
Stratified	25	0.0337	0.0337	0.0266	0.0410	0.0030	101
Systematic	25	0.0341	0.0341	0.0273	0.0418	0.0031	101
Random	50	0.0327	0.0328	0.0269	0.0399	0.0024	101
Stratified	50	0.0326	0.0326	0.0275	0.0393	0.0023	101
Systematic	50	0.0329	0.0328	0.0275	0.0393	0.0024	101
Random	75	0.0323	0.0322	0.0273	0.0379	0.0021	101
Stratified	75	0.0322	0.0321	0.0265	0.0379	0.0020	101
Systematic	75	0.0323	0.0320	0.0272	0.0376	0.0021	101
Random	100	0.0319	0.0318	0.0270	0.0367	0.0019	101
Stratified	100	0.0319	0.0317	0.0274	0.0372	0.0017	101
Systematic	100	0.0319	0.0321	0.0273	0.0369	0.0019	101
Random	200	0.0311	0.0312	0.0276	0.0352	0.0014	101
Stratified	200	0.0311	0.0310	0.0273	0.0358	0.0015	101
Systematic	200	0.0311	0.0309	0.0279	0.0345	0.0014	101
Random	300	0.0307	0.0307	0.0272	0.0342	0.0013	101
Stratified	300	0.0309	0.0309	0.0275	0.0354	0.0014	101
Systematic	300	0.0308	0.0306	0.0266	0.0342	0.0014	101
Random	400	0.0307	0.0306	0.0273	0.0341	0.0011	101
Stratified	400	0.0306	0.0305	0.0268	0.0336	0.0012	101
Systematic	400	0.0307	0.0308	0.0276	0.0336	0.0011	101
Random	500	0.0306	0.0305	0.0279	0.0339	0.0011	101
Stratified	500	0.0307	0.0307	0.0275	0.0337	0.0011	101
Systematic	500	0.0307	0.0308	0.0281	0.0330	0.0009	101

APPENDIX B: Fortran 90 Code for Simulations with Random Sampling

(Note: Array sizes set for image VA15)

```
!     Last change:  RHW  16 Apr 2005   11:55 pm
!
! ksbootfa
!
! Randolph H. Wynne, Department of Forestry, Virginia Polytechnic Institute and State
! University
! Christine E. Blinn, Department of Forestry, Virginia Polytechnic Institute and State
! University
!
! Version 0.3
!
! Program start date: July 29, 2003 (ksboot)
! Inputs:
!       Size of draws
!       Number of random draws (with replacement)
!       Output file root name
!       Input file name for image
!       Number of pixels in the image
!       Number of bands in the image
!       Image (as sequence of brightess value vectors -- row-wise -- as double precision
!        floats)
!       Input file name for accuracy assessment
!       Number of accuracy assessment points
!
! Outputs:
!

program ksbootfa

        USE CEARSwLib
        USE quicksortw_mod

        implicit none


        INTEGER, parameter :: S_LEN = 32
        INTEGER (KIND = 4) :: d, i, j, k, l, p, q, r, s, t, v, w, y, z  ! Loop counters
        INTEGER (KIND = 4) :: BandsLessOne, BandsLessTwo, BandsLessThree, BandsLessFour
        INTEGER (KIND = 4) :: LPlusOne, LPlusOneB, PPlusOne
        INTEGER (KIND = 4) :: JPlusOne, JPlusOneB, JPlusOneC
        INTEGER (KIND = 4) :: IPlusOne, IPlusOneB, IPlusOneC, IPlusOneD
        INTEGER (KIND = 4) :: OneTenthCount, OneTenthNumDraws, SampleNumber, IEdit
        INTEGER (KIND = 4) :: OnePlusJ
        INTEGER (KIND = 4) :: DrawSize
        INTEGER (KIND = 4) :: NumDraws
        INTEGER (KIND = 4) :: HalfDraws
        INTEGER (KIND = 4) :: NumValPoints, NumTrainingPoints
        INTEGER (KIND = 4) :: SeedSize
        INTEGER (KIND = 2) :: a, b, c, e, f, g
        INTEGER (KIND = 4) :: ImageNumNonzeroPixels
        INTEGER (KIND = 4) :: NearestTrain, TrNearestTrain
        INTEGER (KIND = 4) :: TestIndex, TestSpecIndex
        REAL (KIND = 8) :: Xdiff
        REAL (KIND = 8) :: Ydiff
        REAL (KIND = 8) :: TrValXdiff
        REAL (KIND = 8) :: TrValYdiff
        REAL (KIND = 8) :: TrValXYDistW
        REAL (KIND = 8) :: XTrValDiff
        REAL (KIND = 8) :: YTrValDiff
        REAL (KIND = 8) :: TestXDiff
        REAL (KIND = 8) :: TestYDiff
        REAL (KIND = 8) :: TrainXDiff
        REAL (KIND = 8) :: TrainYDiff
        REAL (KIND = 8) :: TrXdiffTr
        REAL (KIND = 8) :: TrYdiffTr
```

```
REAL (KIND = 8) :: TrDistTr
REAL (KIND = 8) :: Acres
REAL (KIND = 8) :: MinXYDist, MinTrXYDist
REAL (KIND = 8) :: TrainValXYDist
REAL (KIND = 8) :: TrainValXDiff
REAL (KIND = 8) :: TrainValYDiff
REAL (KIND = 8) :: TempTrDTr
REAL (KIND = 8) :: TempValDTr
REAL (KIND = 8) :: OnebyOneMatrix (1,1)
REAL (KIND = 8) :: OnebyOneMatrixB (1,1)

INTEGER (KIND = 2) :: InfoClasses (41702)
INTEGER (KIND = 4) :: Locations (41702)
REAL (KIND = 8) :: XYTrain (41702,2)
REAL (KIND = 8) :: XYValData (1742,2)
INTEGER (KIND = 2) :: x (6,98427329)
INTEGER (KIND = 4) :: ValData (1742,2)
REAL (KIND = 8) :: PlotProportionForest (1742)
REAL (KIND = 8) :: RandomNumbers (1000,100)
INTEGER (KIND = 4) :: RandomIndices (1000,100)
REAL (KIND = 8) :: SampleXYTrain (2,500,100)
INTEGER (KIND = 4) :: SampleInfoClasses (500,100)
INTEGER (KIND = 4) :: SampleLocations (500,100)
INTEGER (KIND = 4) :: TestInfoClasses (500,100)
INTEGER (KIND = 4) :: TestLocations (500,100)
REAL (KIND = 8) :: TestXYTrain (2,500,100)
INTEGER (KIND = 4) :: TrainValData (41702,2)
INTEGER (KIND = 4) :: TrainValDataB (41702,2)
REAL (KIND = 8) :: m (6,500,100)
INTEGER (KIND = 2) :: TestVector (6,500,100)
INTEGER (KIND = 2) :: TrainVector (6,41702)
INTEGER (KIND = 2) :: ValVector (6,1742)
REAL (KIND = 8) :: m1 (1,500,100)
INTEGER (KIND = 2) :: x1 (1,1742)
INTEGER (KIND = 4) :: ErrorMatrix1b (6,100,2,2)
REAL (KIND = 8) :: PercentAccuracy1b (6,100)
INTEGER (KIND = 4) :: AAPoints1b (1742,2,6,100)
INTEGER (KIND = 4) :: B1SpecAAPoints (1742,1,6,100)
REAL (KIND = 8) :: B1SpecDistAA (1742,1,6,100)
REAL (KIND = 8) :: m2 (2,500,100)
INTEGER (KIND = 2) :: x2 (2,1742)
INTEGER (KIND = 4) :: ErrorMatrix2b (15,100,2,2)
REAL (KIND = 8) :: PercentAccuracy2b (15,100)
INTEGER (KIND = 4) :: AAPoints2b (1742,2,15,100)
INTEGER (KIND = 4) :: B2SpecAAPoints (1742,1,15,100)
REAL (KIND = 8) :: B2SpecDistAA (1742,1,15,100)
REAL (KIND = 8) :: m3 (3,500,100)
INTEGER (KIND = 2) :: x3 (3,1742)
INTEGER (KIND = 4) :: ErrorMatrix3b (20,100,2,2)
REAL (KIND = 8) :: PercentAccuracy3b (20,100)
INTEGER (KIND = 4) :: AAPoints3b (1742,2,20,100)
INTEGER (KIND = 4) :: B3SpecAAPoints (1742,1,20,100)
REAL (KIND = 8) :: B3SpecDistAA (1742,1,20,100)
REAL (KIND = 8) :: m4 (4,500,100)
INTEGER (KIND = 2) :: x4 (4,1742)
INTEGER (KIND = 4) :: ErrorMatrix4b (15,100,2,2)
REAL (KIND = 8) :: PercentAccuracy4b (15,100)
INTEGER (KIND = 4) :: AAPoints4b (1742,2,15,100)
INTEGER (KIND = 4) :: B4SpecAAPoints (1742,1,15,100)
REAL (KIND = 8) :: B4SpecDistAA (1742,1,15,100)
REAL (KIND = 8) :: m5 (5,500,100)
INTEGER (KIND = 2) :: x5 (5,1742)
INTEGER (KIND = 4) :: ErrorMatrix5b (6,100,2,2)
REAL (KIND = 8) :: PercentAccuracy5b (6,100)
INTEGER (KIND = 4) :: AAPoints5b (1742,2,6,100)
INTEGER (KIND = 4) :: B5SpecAAPoints (1742,1,6,100)
REAL (KIND = 8) :: B5SpecDistAA (1742,1,6,100)
INTEGER (KIND = 4) :: ErrorMatrix6b (100,2,2)
REAL (KIND = 8) :: PercentAccuracy6b (100)
INTEGER (KIND = 4) :: AAPoints6b (1742,2,100)
INTEGER (KIND = 4) :: B6SpecAAPoints (1742,1,100)
```

```
REAL (KIND = 8) :: B6SpecDistAA (1742,1,100)
INTEGER (KIND = 4) :: SortAccuracy (2,100)
INTEGER (KIND = 4) :: WImageDrawNum (11)
REAL (KIND = 8) :: ForestMapMarginal (11)
REAL (KIND = 8) :: StandardError (11)
INTEGER (KIND = 4) :: AAPoints (1742,2,11)
REAL (KIND = 8) :: PrecisionFAE (11)
 INTEGER (KIND = 4) :: TotalClassForest (11)
INTEGER (KIND = 4) :: TotalClassNonforest (11)
REAL (KIND = 8) :: MeanPPForest (11)
REAL (KIND = 8) :: MeanPPNonforest (11)
REAL (KIND = 8) :: WeightedProportion (11)
REAL (KIND = 8) :: ForestArea (11)
REAL (KIND = 8) :: ForestStratVar (11)
REAL (KIND = 8) :: NonforestStratVar (11)
REAL (KIND = 8) :: VarianceSum (11)
REAL (KIND = 8) :: VarianceForestArea (11)
REAL (KIND = 8) :: FAEPrecision (11)
REAL (KIND = 8) :: PercentAccuracy (11)
INTEGER (KIND = 4) :: ErrorMatrix (11,2,2)
REAL (KIND = 8) :: ForestAreaEstimate (11)
REAL (KIND = 8) :: PercentAccuracyTr (11)
INTEGER (KIND = 4) :: ErrorMatrixTr (11,2,2)
REAL (KIND = 8) :: ForestAreaEstimateTr (11)
REAL (KIND = 8) :: StandardErrorTr (11)
INTEGER (KIND = 4) :: AAPointsTr (41702,2,11)
INTEGER (KIND = 4) :: SpecAAPoints (1742,1,11)
REAL (KIND = 8) :: SpecDistAA (1742,1,11)
INTEGER (KIND = 4) :: NumTrPointW (1742,1,11)
REAL (KIND = 8) :: SpatialDistW (1742,1,11)
INTEGER (KIND = 4) :: TestErrorMatrix (100,2,2)
REAL (KIND = 8) :: TestForestAreaEstimate (100)
REAL (KIND = 8) :: TestPercentAccuracy (100)
REAL (KIND = 8) :: TestForestMapMarginal (100)
REAL (KIND = 8) :: TestStandardError (100)
REAL (KIND = 8) :: SpecDistTest (500,1,100)
REAL (KIND = 8) :: TestPrecisionFAE (100)
INTEGER (KIND = 4) :: NumTrPtTest (500,1,100)
INTEGER (KIND = 4) :: NumTrPtSpecTest (500,1,100)
REAL (KIND = 8) :: SpatialDTest (500,1,100)
INTEGER (KIND = 4) :: SpecClassTest (500,1,100)
REAL (KIND = 8) :: TrainPercentAccuracy (100)
REAL (KIND = 8) :: TrainForestMapMarginal (100)
REAL (KIND = 8) :: TrainStandardError (100)
INTEGER (KIND = 4) :: TrainAAPoints (41702,2,100)
INTEGER (KIND = 4) :: TrainErrorMatrix (100,2,2)
REAL (KIND = 8) :: TrainForestAreaEstimate (100)
REAL (KIND = 8) :: SpatialDistTr (41702,1,100)
INTEGER (KIND = 4) :: SpecClassTrain (41702,1,100)
INTEGER (KIND = 4) :: NumTrPtTrain (41702,1,100)
REAL (KIND = 8) :: SpecDistTrain (41702,1,100)
REAL (KIND = 8) :: TrainPrecisionFAE (100)
INTEGER (KIND = 4) :: ForestSampleSize (100)
INTEGER (KIND = 4) :: NonforestSampleSize (100)
INTEGER (KIND = 4) :: TrMinXY2TrainA (500,2,100)
REAL (KIND = 8) :: TrMinXY2Train (500,2,100)
REAL (KIND = 8) :: TrminusTr (6,1)
INTEGER (KIND = 4) :: ValMinXY2TrainA (1742,1,100)
REAL (KIND = 8) :: ValMinXY2Train (1742,2,100)
REAL (KIND = 8) :: ValminusTr (6,1)

INTEGER, ALLOCATABLE :: Seed (:)
INTEGER (KIND = 4), ALLOCATABLE :: TestValData (:,:,:)
INTEGER (KIND = 4), ALLOCATABLE :: TestAAPoints (:,:,:)
INTEGER (KIND = 2), ALLOCATABLE :: scx (:), icx (:)
INTEGER (KIND = 2), ALLOCATABLE :: Testscx (:), Testicx (:)
INTEGER (KIND = 2), ALLOCATABLE :: Trainscx (:), Trainicx (:)

REAL (KIND = 8), ALLOCATABLE :: DistanceImage (:)
REAL (KIND = 8), ALLOCATABLE :: TestDistImage (:)
REAL (KIND = 8), ALLOCATABLE :: TrainDistImage (:)
```

```
INTEGER (KIND = 2), ALLOCATABLE :: scx1 (:), icx1 (:)
INTEGER (KIND = 2), ALLOCATABLE :: scx2 (:), icx2 (:)
INTEGER (KIND = 2), ALLOCATABLE :: scx3 (:), icx3 (:)
INTEGER (KIND = 2), ALLOCATABLE :: scx4 (:), icx4 (:)
INTEGER (KIND = 2), ALLOCATABLE :: scx5 (:), icx5 (:)
INTEGER (KIND = 2), ALLOCATABLE :: scx6 (:), icx6 (:)
REAL (KIND = 8), ALLOCATABLE :: DistImage1B (:)
REAL (KIND = 8), ALLOCATABLE :: DistImage2B (:)
REAL (KIND = 8), ALLOCATABLE :: DistImage3B (:)
REAL (KIND = 8), ALLOCATABLE :: DistImage4B (:)
REAL (KIND = 8), ALLOCATABLE :: DistImage5B (:)
REAL (KIND = 8), ALLOCATABLE :: DistImage6B (:)


CHARACTER (LEN = S_LEN) :: InputFileNameImage
CHARACTER (LEN = S_LEN) :: InputFileNameTrainingData
CHARACTER (LEN = S_LEN) :: InputFileNameTrainingXY
CHARACTER (LEN = S_LEN) :: InputFileNameValidationData
CHARACTER (LEN = S_LEN) :: InputFileNamePlotProportion
CHARACTER (LEN = S_LEN) :: InputFileNameValDataXY
CHARACTER (LEN = S_LEN) :: OutputFileNameRoot
CHARACTER (LEN = S_LEN) :: OutputFileNameValuesAll
CHARACTER (LEN = S_LEN) :: OutputFileNameKSAll
CHARACTER (LEN = S_LEN) :: OutputFileNameMeanVectors
CHARACTER (LEN = S_LEN) :: OutputFileNameAAPoints
CHARACTER (LEN = S_LEN) :: OutputFileNameXYDistance
CHARACTER (LEN = S_LEN) :: OutputFileNameTrainDistA
CHARACTER (LEN = S_LEN) :: OutputFileNameTrainDistB
CHARACTER (LEN = S_LEN) :: OutputFileNameWAATrain
CHARACTER (LEN = S_LEN) :: OutputFileNameAATrain
CHARACTER (LEN = S_LEN) :: OutputFileNameTestDist
CHARACTER (LEN = S_LEN) :: OutputFileNameAATest
CHARACTER (LEN = S_LEN) :: OutputFileNameTestVectors
CHARACTER (LEN = S_LEN) :: OutputFileNameAreaEst2
CHARACTER (LEN = S_LEN) :: OutputFileNameTrValXYDist
CHARACTER (LEN = S_LEN) :: OutputFileNameSpecDistTrain
CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo1AA
CHARACTER (LEN = S_LEN) :: OutputFileNameB1AAPoints
CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo2AA
CHARACTER (LEN = S_LEN) :: OutputFileNameB2AAPoints
CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo3AA
CHARACTER (LEN = S_LEN) :: OutputFileNameB3AAPoints
CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo4AA
CHARACTER (LEN = S_LEN) :: OutputFileNameB4AAPoints
CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo5AA
CHARACTER (LEN = S_LEN) :: OutputFileNameB5AAPoints
CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo6AA
CHARACTER (LEN = S_LEN) :: OutputFileNameB6AAPoints
CHARACTER (LEN = S_LEN) :: OutputFileNameSpecDValTr
CHARACTER (LEN = S_LEN) :: OutputFileNameSortAcc
CHARACTER (LEN = S_LEN) :: OutputFileNameTrMinXYTr
CHARACTER (LEN = S_LEN) :: OutputFileNameValMinXYTr
CHARACTER (LEN = S_LEN) :: OutputFileNameValVectors
CHARACTER (LEN = S_LEN) :: OutputFileNameTrainVectors

       open (10, file='15m_random.in', status = "old", action = "read")

       read (10, *) DrawSize
       PRINT *, "Draw Size: ", DrawSize

       read (10, *) NumDraws
        PRINT *, "Number of Draws: ", NumDraws

       read (10, *) InputFileNameImage
        PRINT *, "Input File Name Raw Image: ", InputFileNameImage

       read (10, *) k
       PRINT *, "Number of Pixels in Raw Image: ", k

       read (10, *) b
```

```
        PRINT *, "Number of Bands: ", b

       read (10, *) ImageNumNonzeroPixels
      PRINT *, "Number of Nonzero Pixels in Raw Image: ", ImageNumNonzeroPixels

       read (10, *) InputFileNameTrainingData
        PRINT *, "Input File Name Training Data: ", InputFileNameTrainingData

       read (10, *) InputFileNameTrainingXY
        PRINT *, "Input File Name Training XY: ", InputFileNameTrainingXY

       read (10, *) NumTrainingPoints
        PRINT *, "Number of Training Points: ", NumTrainingPoints

       read (10, *) InputFileNameValidationData
        PRINT *, "Input File Name Validation Data: ", InputFileNameValidationData

       read (10, *) InputFileNameValDataXY
       PRINT *, "Input File Name Validation XY: ", InputFileNameValDataXY

       read (10, *) InputFileNamePlotProportion
       PRINT *, "Input File Name Validation Data: ", InputFileNamePlotProportion

       read (10, *) NumValPoints
       PRINT *, "Number of Validation Points: ",  NumValPoints

       read (10, *) OutputFileNameRoot
       PRINT *, "Output File Name Root (no suffix, e.g. 1000x1000): ", &
              OutputFileNameRoot


OutputFileNameValuesAll = TRIM (OutputFileNameRoot) // "Values_All.txt"
OutputFileNameKSAll = TRIM (OutputFileNameRoot) // "KS_All.txt"
OutputFileNameMeanVectors = TRIM (OutputFileNameRoot) // "Mean_Vectors.txt"
 OutputFileNameAAPoints = TRIM (OutputFileNameRoot) // "AAPoints.txt"
OutputFileNameXYDistance = TRIM (OutputFileNameRoot) // "XYDistance.txt"
OutputFileNameTrainDistA = TRIM (OutputFileNameRoot) // "TrainDistA.txt"
OutputFileNameTrainDistB = TRIM (OutputFileNameRoot) // "TrainDistB.txt"
OutputFileNameWAATrain = TRIM (OutputFileNameRoot) // "WAATrain.txt"
OutputFileNameAATrain = TRIM (OutputFileNameRoot) // "AATrain.txt"
OutputFileNameTestDist = TRIM (OutputFileNameRoot) // "TestDist.txt"
OutputFileNameAATest = TRIM (OutputFileNameRoot) // "AATest.txt"
OutputFileNameTestVectors = TRIM (OutputFileNameRoot) // "Test_Vectors.txt"
OutputFileNameAreaEst2 = TRIM (OutputFileNameRoot) // "AreaEst2.txt"
OutputFileNameTrValXYDist = TRIM (OutputFileNameRoot) // "TrValXYDist.txt"
OutputFileNameSpecDistTrain = TRIM (OutputFileNameRoot) // "SpecDistTrain.txt"
OutputFileNameBandCombo1AA = TRIM (OutputFileNameRoot) // "BandCombo1AA.txt"
OutputFileNameB1AAPoints = TRIM (OutputFileNameRoot) // "B1AAPoints.txt"
OutputFileNameBandCombo2AA = TRIM (OutputFileNameRoot) // "BandCombo2AA.txt"
OutputFileNameB2AAPoints = TRIM (OutputFileNameRoot) // "B2AAPoints.txt"
OutputFileNameBandCombo3AA = TRIM (OutputFileNameRoot) // "BandCombo3AA.txt"
OutputFileNameB3AAPoints = TRIM (OutputFileNameRoot) // "B3AAPoints.txt"
OutputFileNameBandCombo4AA = TRIM (OutputFileNameRoot) // "BandCombo4AA.txt"
OutputFileNameB4AAPoints = TRIM (OutputFileNameRoot) // "B4AAPoints.txt"
OutputFileNameBandCombo5AA = TRIM (OutputFileNameRoot) // "BandCombo5AA.txt"
OutputFileNameB5AAPoints = TRIM (OutputFileNameRoot) // "B5AAPoints.txt"
OutputFileNameBandCombo6AA = TRIM (OutputFileNameRoot) // "BandCombo6AA.txt"
OutputFileNameB6AAPoints = TRIM (OutputFileNameRoot) // "B6AAPoints.txt"
OutputFileNameSpecDValTr = TRIM (OutputFileNameRoot) // "SpecDValTr.txt"
OutputFileNameSortAcc = TRIM (OutputFileNameRoot) // "SortAcc.txt"
OutputFileNameTrMinXYTr = TRIM (OutputFileNameRoot) // "TrMinXYTr.txt"
OutputFileNameValMinXYTr = TRIM (OutputFileNameRoot) // "ValMinXYTr.txt"
OutputFileNameValVectors = TRIM (OutputFileNameRoot) // "Val_Vectors.txt"
OutputFileNameTrainVectors = TRIM (OutputFileNameRoot) // "Train_Vectors.txt"

write (unit = *, FMT = *) "Input file names: "
write (unit = *, FMT = *) "    ", InputFileNameImage
write (unit = *, FMT = *) "    ", InputFileNameTrainingData
write (unit = *, FMT = *) "    ", InputFileNameValidationData
write (unit = *, FMT = *) "    ", InputFileNamePlotProportion
write (unit = *, FMT = *) "    ", InputFileNameTrainingXY
```

```
write (unit = *, FMT = *) "    ", InputFileNameValDataXY

write (unit = *, FMT = *) "Draw Size: "
write (unit = *, FMT = *) "    ", DrawSize

write (unit = *, FMT = *) "Number of Draws: "
write (unit = *, FMT = *) "    ", NumDraws

write (unit = *, FMT = *) "Number of Bands: "
write (unit = *, FMT = *) "    ", b, "(Raw Image)"

write (unit = *, FMT = *) "Number of Pixels in raw image: "
write (unit = *, FMT = *) "    ", k, " (Total)"

write (unit = *, FMT = *) "    ", ImageNumNonzeroPixels, " (Nonzero)"

write (unit = *, FMT = *) "Number of Validation Points: "
write (unit = *, FMT = *) "    ", NumValPoints

write (unit = *, FMT = *) "Output file names: "
write (unit = *, FMT = *) "    ", OutputFileNameValuesAll
write (unit = *, FMT = *) "    ", OutputFileNameKSAll
write (unit = *, FMT = *) "    ", OutputFileNameMeanVectors
write (unit = *, FMT = *) "    ", OutputFileNameAAPoints
write (unit = *, FMT = *) "    ", OutputFileNameXYDistance
write (unit = *, FMT = *) "    ", OutputFileNameTrainDistA
write (unit = *, FMT = *) "    ", OutputFileNameTrainDistB
write (unit = *, FMT = *) "    ", OutputFileNameWAATrain
write (unit = *, FMT = *) "    ", OutputFileNameAATrain
write (unit = *, FMT = *) "    ", OutputFileNameTestDist
write (unit = *, FMT = *) "    ", OutputFileNameAATest
write (unit = *, FMT = *) "    ", OutputFileNameTestVectors
write (unit = *, FMT = *) "    ", OutputFileNameAreaEst2
write (unit = *, FMT = *) "    ", OutputFileNameTrValXYDist
write (unit = *, FMT = *) "    ", OutputFileNameSpecDistTrain
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo1AA
write (unit = *, FMT = *) "    ", OutputFileNameB1AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo2AA
write (unit = *, FMT = *) "    ", OutputFileNameB2AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo3AA
write (unit = *, FMT = *) "    ", OutputFileNameB3AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo4AA
write (unit = *, FMT = *) "    ", OutputFileNameB4AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo5AA
write (unit = *, FMT = *) "    ", OutputFileNameB5AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo6AA
write (unit = *, FMT = *) "    ", OutputFileNameB6AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameSpecDValTr
write (unit = *, FMT = *) "    ", OutputFileNameSortAcc
write (unit = *, FMT = *) "    ", OutputFileNameTrMinXYTr
write (unit = *, FMT = *) "    ", OutputFileNameValMinXYTr
write (unit = *, FMT = *) "    ", OutputFileNameValVectors
write (unit = *, FMT = *) "    ", OutputFileNameTrainVectors

r = 2 * DrawSize
s = DrawSize + 1
OneTenthNumDraws = ((NumDraws/10) + 1)

 PRINT *, "OneTenthNumDraws = ", OneTenthNumDraws

TempTrDTr = 0.0
TempValDTr = 0.0
TestIndex = 0
TestSpecIndex = 0
Acres = 0.0
Xdiff = 0.0
Ydiff = 0.0
 TrainValXDiff = 0.0
TrainValYDiff = 0.0
TrValXdiff = 0.0
TrValYdiff = 0.0
```

```
TrValXYDistW = 0.0
TrXdiffTr = 0.0
TrYdiffTr = 0.0
TrainXDiff = 0.0
TrainYDiff = 0.0
TestXDiff = 0.0
TestYDiff = 0.0
XTrValDiff = 0.0
YTrValDiff = 0.0
BandsLessOne = 0
BandsLessTwo = 0
BandsLessThree = 0
BandsLessFour = 0
IPlusOne = 0
IPlusOneB = 0
IPlusOneC = 0
IPlusOneD = 0
JPlusOne = 0
JPlusOneB = 0
JPlusOneC = 0
LPlusOne = 0
LPlusOneB = 0
PPlusOne = 0
OnePlusJ = 0
NearestTrain = 0
NonforestStratVar = 0.0
ForestStratVar = 0.0
MeanPPForest = 0.0
MeanPPNonforest = 0.0
HalfDraws = 0

open (unit = 1, file = InputFileNameTrainingData, access = "sequential", &
       form = "formatted")

do i = 1, NumTrainingPoints

        read (unit = 1, fmt = "(I1)") InfoClasses(i)

end do

PRINT *, "Information classes read."


do i = 1, NumTrainingPoints

        read (unit = 1, fmt = "(I8)") Locations (i)

end do

close (unit = 1)

PRINT *, "Locations read."


open (unit = 22, file = InputFileNameTrainingXY, access = "sequential", &
       form = "formatted")

do i = 1, NumTrainingPoints

        read (unit = 22, fmt = "(F18.9)" ) XYTrain (i,1)

end do

do i = 1, NumTrainingPoints

        read (unit = 22, fmt = "(F18.9)" ) XYTrain (i,2)

end do

close (unit = 22)
```

```
PRINT *, "X and Y of training points read."


open (unit = 31, file = InputFileNameValDataXY, access = "sequential", &
       form = "formatted")

do i = 1, NumValPoints

        read (unit = 31, fmt = "(F18.9)" ) XYValData (i,1)

end do

do i = 1, NumValPoints

        read (unit = 31, fmt = "(F18.9)" ) XYValData (i,2)

end do

close (unit = 31)

PRINT *, "X and Y of validation points read."


open (unit = 11, file = InputFileNameImage, &
       form = "binary", status = "old", action = "read")

write (unit = *, fmt = *) "Reading input image into memory..."

ReadRawPixelsLoop: do i = 1, k

       ReadRawBandsLoop: do j = 1, b

               read (unit = 11) x (j,i) !, fmt = "(I)")

    end do ReadRawBandsLoop

end do ReadRawPixelsLoop


close (unit = 11)

write (unit = *, fmt = *) "Raw image written into memory..."


open (unit = 12, file = InputFileNameValidationData, access = "sequential", &
       form = "formatted")

PRINT *, "Reading validation data..."

ReadValidationValuesLoop: do i = 1, NumValPoints

        read (unit = 12, fmt = *) ValData (i,1)

end do ReadValidationValuesLoop

ReadValidationLocationsLoop: do i = 1, NumValPoints

        read (unit = 12, fmt = *) ValData (i,2)

end do ReadValidationLocationsLoop


close (unit = 12)

PRINT *, "Reading plot proportions..."

PlotProportionForest = 0.0

open (unit = 30, file = InputFileNamePlotProportion, access = "sequential", &
       form = "formatted")
```

```
ReadPlotProportionFLoop: do i = 1, NumValPoints

        read (unit = 30, fmt = "(F12.10)")  PlotProportionForest (i)

end do ReadPlotProportionFLoop

close (unit = 30)

PRINT *, "Generating random numbers and random indices..."

call random_seed ()

call random_seed (size = SeedSize)
ALLOCATE (Seed(SeedSize))
call random_seed (get = Seed)
PRINT *, 'Random seed = ', Seed

CALL random_number (RandomNumbers)


PRINT *, "First Random Number is: ", RandomNumbers (1,1)

RandomIndices = int (RandomNumbers * NumTrainingPoints + 1, 4)


PRINT *, "Creating Samples, SampleInfoClasses, and SampleLocations..."

CreateSamples: do i = 1, NumDraws

        do j = 1, DrawSize

               SampleInfoClasses (j,i) = InfoClasses (RandomIndices (j,i))
                SampleLocations (j,i) = Locations (RandomIndices (j,i) )
                SampleXYTrain (:,j,i) = XYTrain (RandomIndices (j,i), : )

               end do

        do l = s, r

               TestInfoClasses (l-DrawSize,i) = InfoClasses (RandomIndices (l,i))
                TestLocations (l-DrawSize,i) = Locations (RandomIndices (l,i) )
                TestXYTrain (:,l-DrawSize,i) = XYTrain (RandomIndices (l,i), : )

        end do

end do CreateSamples


PRINT *, "Creating test validation data..."

ALLOCATE (TestValData (DrawSize,2,NumDraws))

TestValDataLoop: do i = 1, NumDraws

        do j = 1, DrawSize

       TestValData (j,1,i) = TestInfoClasses (j,i)
       TestValData (j,2,i) = j

        end do

end do TestValDataLoop

PRINT *, "Creating train validation data..."

TrainValDataLoop: do i = 1, NumTrainingPoints

       TrainValData (i,1) = InfoClasses (i)
       TrainValData (i,2) = Locations (i)

end do TrainValDataLoop
```

```
  TrainValDataBLoop: do i = 1, NumTrainingPoints

         TrainValDataB (i,1) = InfoClasses (i)
         TrainValDataB (i,2) = i

  end do TrainValDataBLoop


PRINT *, "Creating mean vectors"

  OuterMeanVectorsLoop: do i = 1, NumDraws

      InnerMeanVectorsLoop: do j = 1, DrawSize

          m (:, j, i) = x (:,SampleLocations (j,i))

      end do InnerMeanVectorsLoop

  end do OuterMeanVectorsLoop

  PRINT *, "Creating test vectors..."

  OuterTestVectorsLoop: do i = 1, NumDraws

      InnerTestVectorsLoop: do j = 1, DrawSize

          TestVector (:, j, i) = x (:, TestLocations (j,i))

      end do InnerTestVectorsLoop

  end do OuterTestVectorsLoop

  PRINT *, "Creating train vectors..."

  TrainingVectorsLoop: do i = 1, NumTrainingPoints

          TrainVector (:, i) = x (:, Locations (i))

  end do TrainingVectorsLoop

  open (unit = 50, file = OutputFileNameTrainVectors, access = "sequential", &
      form = "formatted", action = "write", status = "replace")

     WriteTrainVectors: do j = 1, NumTrainingPoints

          write (unit = 50,  fmt = "(I10, I10, I10, I10, I10, I10)") &
                TrainVector (:,j)

      end do WriteTrainVectors

  close (unit = 50)

  PRINT *, "Creating validation vector..."

  ValidationVectorsLoop: do i = 1, NumValPoints

          ValVector (:, i) = x (:, ValData (i,2))

  end do ValidationVectorsLoop

  open (unit = 49,  file = OutputFileNameValVectors, access = "sequential", &
      form = "formatted", action = "write", status = "replace")

     WriteValVectors: do j = 1, NumValPoints

          write (unit = 49,  fmt = "(I10, I10, I10, I10, I10, I10)") &
                ValVector (:,j)

      end do WriteValVectors
```

```
        close (unit = 49)

        Acres = (0.22239 * ImageNumNonzeroPixels)

        PRINT *, "Acres = ", Acres
        PRINT *, "Beginning band combinations, 1 band loop..."

! Exploring best subset of bands with all possible band combinations using just the
! validation data

       a = 1

        ErrorMatrix1b = 0
        PercentAccuracy1b = 0.0
        B1SpecAAPoints = 0
        B1SpecDistAA = 0.0

       OneBandLoop: do i = 1, b

               m1(1,:,:) = m(i,:,:)
               x1(1,:) = ValVector(i,:)


                      Classification1BLoop: do d = 1, NumDraws

                     ALLOCATE (scx1 (NumValPoints))
                     ALLOCATE (icx1 (NumValPoints))
                     ALLOCATE (DistImage1B (NumValPoints))

                     scx1 = 0
                     icx1 = 0
                      DistImage1B = 0.0

                       call nn (a, NumValPoints, DrawSize, x1, m1(:,:,d), scx1, &
                            DistImage1B)

                       call Recode (NumValPoints, scx1, SampleInfoClasses (:,d), icx1)

                       call ComputeAccuracyOnly (NumValPoints, icx1, NumValPoints, &
                            ValData, ErrorMatrix1b(i,d,:,:), PercentAccuracy1b(i,d), &
                            AAPoints1b(:,:,i,d))


                     do t = 1, NumValPoints

                             B1SpecAAPoints (t,1,i,d) = scx1 (t)
                             B1SpecDistAA (t,1,i,d) = DistImage1B (t)

                     end do

                     DEALLOCATE (scx1)
                     DEALLOCATE (icx1)
                     DEALLOCATE (DistImage1B)

                      end do Classification1BLoop

                      m1 = 0.0
                      x1 = 0

        end do OneBandLoop


        open (unit = 32,  file = OutputFileNameBandCombo1AA, access = "sequential", &
            form = "formatted", action = "write", status = "replace")

        open (unit = 33,  file = OutputFileNameB1AAPoints, access = "sequential", &
            form = "formatted", action = "write", status = "replace")

        WriteB1Files: do i = 1, NumDraws
```

```
        WriteBandCombo1: do j = 1, 6

            write (unit = 32, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
           ErrorMatrix1b (j,i,1,1), ErrorMatrix1b (j,i,1,2), &
                  ErrorMatrix1b (j,i,2,1), ErrorMatrix1b (j,i,2,2), &
                  PercentAccuracy1b (j,i)

                  end do WriteBandCombo1

        WriteB1ComboAAPoints: do l = 1, 6

            do j = 1, NumValPoints

            write (unit = 33,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
            AAPoints1b (j,1,l,i), AAPoints1b (j,2,l,i), &
           B1SpecAAPoints (j,1,l,i), B1SpecDistAA (j,1,l,i)

            end do

        end do WriteB1ComboAAPoints

       end do WriteB1Files

close (unit = 32)
close (unit = 33)


    PRINT *, "Beginning band combinations, 2 band loop..."

    z = 0
    c = 2

    BandsLessOne = b - 1

    ErrorMatrix2b = 0
    PercentAccuracy2b = 0.0
    B2SpecAAPoints = 0
    B2SpecDistAA = 0.0

   TwoBandLoop: do i = 1, BandsLessOne

                   IPlusOne = i + 1

                   Band2of2Loop: do j = IPlusOne, b

                          m2(1,:,:) = m(i,:,:)
                          m2(2,:,:) = m(j,:,:)
                         x2(1,:) = ValVector(i,:)
                         x2(2,:) = ValVector(j,:)

                   z = z + 1

                   Classification2BLoop: do d = 1, NumDraws

                  ALLOCATE (scx2 (NumValPoints))
                  ALLOCATE (icx2 (NumValPoints))
                  ALLOCATE (DistImage2B (NumValPoints))

                  scx2 = 0
                  icx2 = 0
                   DistImage2B = 0.0

                    call nn (c, NumValPoints, DrawSize, x2, m2(:,:,d), scx2, &
                  DistImage2B)

                    call Recode (NumValPoints, scx2, SampleInfoClasses (:,d), icx2)

                    call ComputeAccuracyOnly (NumValPoints, icx2, NumValPoints, &
                  ValData, ErrorMatrix2b(z,d,:,:), PercentAccuracy2b(z,d), &
          AAPoints2b(:,:,z,d))
```

```
                do t = 1, NumValPoints

                        B2SpecAAPoints (t,1,z,d) = scx2 (t)
                        B2SpecDistAA (t,1,z,d) = DistImage2B (t)

                end do

                DEALLOCATE (scx2)
                DEALLOCATE (icx2)
                DEALLOCATE (DistImage2B)

                 end do Classification2BLoop

                 m2 = 0.0
                 x2 = 0

                 end do Band2of2Loop

    end do TwoBandLoop


    open (unit = 34,  file = OutputFileNameBandCombo2AA, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 35,  file = OutputFileNameB2AAPoints, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteB2Files: do i = 1, NumDraws

        WriteBandCombo2: do j = 1, 15

         write (unit = 34, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
        ErrorMatrix2b (j,i,1,1), ErrorMatrix2b (j,i,1,2), &
               ErrorMatrix2b (j,i,2,1), ErrorMatrix2b (j,i,2,2),&
               PercentAccuracy2b (j,i)

               end do WriteBandCombo2

        WriteB2ComboAAPoints: do l = 1, 15

         do j = 1, NumValPoints

         write (unit = 35,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
               AAPoints2b (j,1,l,i), AAPoints2b (j,2,l,i), &
        B2SpecAAPoints (j,1,l,i), B2SpecDistAA (j,1,l,i)

         end do

        end do WriteB2ComboAAPoints

       end do WriteB2Files

close (unit = 34)
close (unit = 35)


    PRINT *, "Beginning band combinations, 3 band loop..."

    y = 0
    e = 3

    BandsLessTwo = b - 2

    ErrorMatrix3b = 0
    PercentAccuracy3b = 0.0
    B3SpecAAPoints = 0
    B3SpecDistAA = 0.0

   ThreeBandLoop: do i = 1, BandsLessTwo
```

```
    IPlusOneB = i + 1

    Band2of3Loop: do j = IPlusOneB, BandsLessOne

        JPlusOne = j + 1

        Band3of3Loop: do l = JPlusOne, b

           m3(1,:,:) = m(i,:,:)
           m3(2,:,:) = m(j,:,:)
           m3(3,:,:) = m(l,:,:)
            x3(1,:) = ValVector(i,:)
            x3(2,:) = ValVector(j,:)
            x3(3,:) = ValVector(l,:)

                y = y + 1

                Classification3BLoop: do d = 1, NumDraws

              ALLOCATE (scx3 (NumValPoints))
              ALLOCATE (icx3 (NumValPoints))
              ALLOCATE (DistImage3B (NumValPoints))

               scx3 = 0
               icx3 = 0
              DistImage3B = 0.0

                 call nn (e, NumValPoints, DrawSize, x3, m3(:,:,d), scx3, &
                      DistImage3B)

                 call Recode (NumValPoints, scx3, SampleInfoClasses (:,d), icx3)

                 call ComputeAccuracyOnly (NumValPoints, icx3, NumValPoints, &
                       ValData, ErrorMatrix3b(y,d,:,:), PercentAccuracy3b(y,d), &
                       AAPoints3b(:,:,y,d))


               do t = 1, NumValPoints

                       B3SpecAAPoints (t,1,y,d) = scx3 (t)
                       B3SpecDistAA (t,1,y,d) = DistImage3B (t)

              end do

              DEALLOCATE (scx3)
              DEALLOCATE (icx3)
              DEALLOCATE (DistImage3B)

                end do Classification3BLoop

                m3 = 0.0
                x3 = 0

        end do Band3of3Loop

    end do Band2of3Loop

end do ThreeBandLoop


open (unit = 36,  file = OutputFileNameBandCombo3AA, access = "sequential", &
    form = "formatted", action = "write", status = "replace")

open (unit = 37,  file = OutputFileNameB3AAPoints, access = "sequential", &
    form = "formatted", action = "write", status = "replace")

WriteB3Files: do i = 1, NumDraws

    WriteBandCombo3: do j = 1, 20

        write (unit = 36, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
```

```
                    ErrorMatrix3b (j,i,1,1), ErrorMatrix3b (j,i,1,2), &
                    ErrorMatrix3b (j,i,2,1), ErrorMatrix3b (j,i,2,2),&
                    PercentAccuracy3b (j,i)

                  end do WriteBandCombo3

        WriteB3ComboAAPoints: do l = 1, 20

            do j = 1, NumValPoints

            write (unit = 37,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
            AAPoints3b (j,1,l,i), AAPoints3b (j,2,l,i), &
           B3SpecAAPoints (j,1,l,i), B3SpecDistAA (j,1,l,i)

            end do

        end do WriteB3ComboAAPoints

       end do WriteB3Files

close (unit = 36)
close (unit = 37)


    PRINT *, "Beginning band combinations, 4 band loop..."

    w = 0
    f = 4

    BandsLessThree = b - 3

    ErrorMatrix4b = 0
    PercentAccuracy4b = 0.0
    B4SpecAAPoints = 0
    B4SpecDistAA = 0.0

   FourBandLoop: do i = 1, BandsLessThree

        IPlusOneC = i + 1

        Band2of4Loop: do j = IPlusOneC, BandsLessTwo

            JPlusOneB = j + 1

            Band3of4Loop: do l = JPlusOneB, BandsLessOne

                LPlusOne = l + 1

                Band4of4Loop: do p = LPlusOne, b

               m4(1,:,:) = m(i,:,:)
               m4(2,:,:) = m(j,:,:)
               m4(3,:,:) = m(l,:,:)
               m4(4,:,:) = m(p,:,:)
                x4(1,:) = ValVector(i,:)
                x4(2,:) = ValVector(j,:)
                x4(3,:) = ValVector(l,:)
                x4(4,:) = ValVector(p,:)

                    w = w + 1

                    Classification4BLoop: do d = 1, NumDraws

                  ALLOCATE (scx4 (NumValPoints))
                  ALLOCATE (icx4 (NumValPoints))
                  ALLOCATE (DistImage4B (NumValPoints))

                   scx4 = 0
            icx4 = 0
                 DistImage4B = 0.0
```

```
                        call nn (f, NumValPoints, DrawSize, x4, m4(:,:,d), scx4, &
                              DistImage4B)

                        call Recode (NumValPoints, scx4, SampleInfoClasses (:,d), icx4)

                        call ComputeAccuracyOnly (NumValPoints, icx4, NumValPoints, &
                              ValData, ErrorMatrix4b(w,d,:,:), PercentAccuracy4b(w,d), &
                              AAPoints4b(:,:,w,d))


                       do t = 1, NumValPoints

                              B4SpecAAPoints (t,1,w,d) = scx4 (t)
                              B4SpecDistAA (t,1,w,d) = DistImage4B (t)

                       end do

                       DEALLOCATE (scx4)
                       DEALLOCATE (icx4)
                       DEALLOCATE (DistImage4B)

                        end do Classification4BLoop

                        m4 = 0.0
                        x4 = 0

                    end do Band4of4Loop

                end do Band3of4Loop

            end do Band2of4Loop

        end do FourBandLoop

        open (unit = 38,  file = OutputFileNameBandCombo4AA, access = "sequential", &
            form = "formatted", action = "write", status = "replace")

        open (unit = 39,  file = OutputFileNameB4AAPoints, access = "sequential", &
            form = "formatted", action = "write", status = "replace")

        WriteB4Files: do i = 1, NumDraws

            WriteBandCombo4: do j = 1, 15

                write (unit = 38, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
                      ErrorMatrix4b (j,i,1,1), ErrorMatrix4b (j,i,1,2), &
                      ErrorMatrix4b (j,i,2,1), ErrorMatrix4b (j,i,2,2),&
               PercentAccuracy4b (j,i)

                      end do WriteBandCombo4

            WriteB4ComboAAPoints: do l = 1, 15

                do j = 1, NumValPoints

                write (unit = 39,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
                AAPoints4b (j,1,l,i), AAPoints4b (j,2,l,i), &
               B4SpecAAPoints (j,1,l,i), B4SpecDistAA (j,1,l,i)

                end do

            end do WriteB4ComboAAPoints

           end do WriteB4Files

    close (unit = 38)
    close (unit = 39)


        PRINT *, "Beginning band combinations, 5 band loop..."
```

```
v = 0
g = 5

BandsLessFour = b - 4
ErrorMatrix5b = 0
PercentAccuracy5b = 0.0
B5SpecAAPoints = 0
B5SpecDistAA = 0.0

FiveBandLoop: do i = 1, BandsLessFour

    IPlusOneD = i + 1

    Band2of5Loop: do j = IPlusOneD, BandsLessThree

        JPlusOneC = j + 1

        Band3of5Loop: do l = JPlusOneC, BandsLessTwo

            LPlusOneB = l + 1

            Band4of5Loop: do p = LPlusOneB, BandsLessOne

                PPlusOne = p + 1

                Band5of5Loop: do q = PPlusOne, b

                   m5(1,:,:) = m(i,:,:)
                   m5(2,:,:) = m(j,:,:)
                   m5(3,:,:) = m(l,:,:)
                   m5(4,:,:) = m(p,:,:)
                   m5(5,:,:) = m(q,:,:)
                    x5(1,:) = ValVector(i,:)
                    x5(2,:) = ValVector(j,:)
                    x5(3,:) = ValVector(l,:)
                    x5(4,:) = ValVector(p,:)
                    x5(5,:) = ValVector(q,:)

                v = v + 1

                Classification5BLoop: do d = 1, NumDraws

              ALLOCATE (scx5 (NumValPoints))
              ALLOCATE (icx5 (NumValPoints))
              ALLOCATE (DistImage5B (NumValPoints))

               scx5 = 0
               icx5 = 0
             DistImage5B = 0.0

                call nn (g, NumValPoints, DrawSize, x5, m5(:,:,d), scx5, &
             DistImage5B)

                call Recode (NumValPoints, scx5, SampleInfoClasses (:,d), icx5)

                call ComputeAccuracyOnly (NumValPoints, icx5, NumValPoints, &
                      ValData, ErrorMatrix5b(v,d,:,:), PercentAccuracy5b(v,d), &
                      AAPoints5b(:,:,v,d))


              do t = 1, NumValPoints

                      B5SpecAAPoints (t,1,v,d) = scx5 (t)
                      B5SpecDistAA (t,1,v,d) = DistImage5B (t)

              end do

              DEALLOCATE (scx5)
              DEALLOCATE (icx5)
              DEALLOCATE (DistImage5B)
```

```
                    end do Classification5BLoop

                    m5 = 0.0
                    x5 = 0

                    end do Band5of5Loop

                end do Band4of5Loop

            end do Band3of5Loop

        end do Band2of5Loop

     end do FiveBandLoop


     open (unit = 40,  file = OutputFileNameBandCombo5AA, access = "sequential", &
         form = "formatted", action = "write", status = "replace")

     open (unit = 41,  file = OutputFileNameB5AAPoints, access = "sequential", &
         form = "formatted", action = "write", status = "replace")

    WriteB5Files: do i = 1, NumDraws

         WriteBandCombo5: do j = 1, 6

             write (unit = 40, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
                   ErrorMatrix5b (j,i,1,1), ErrorMatrix5b (j,i,1,2), &
                    ErrorMatrix5b (j,i,2,1), ErrorMatrix5b (j,i,2,2),&
             PercentAccuracy5b (j,i)

                   end do WriteBandCombo5

         WriteB5ComboAAPoints: do l = 1, 6

            do j = 1, NumValPoints

             write (unit = 41,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
             AAPoints5b (j,1,l,i), AAPoints5b (j,2,l,i), &
            B5SpecAAPoints (j,1,l,i), B5SpecDistAA (j,1,l,i)

             end do

         end do WriteB5ComboAAPoints

        end do WriteB5Files

close (unit = 40)
close (unit = 41)


     PRINT *, "Beginning band combinations, 6 band loop..."

    ErrorMatrix6b = 0
    PercentAccuracy6b = 0.0
    B6SpecAAPoints = 0
    B6SpecDistAA = 0.0

    Classification6BLoop: do d = 1, NumDraws


    ALLOCATE (scx6 (NumValPoints))
    ALLOCATE (icx6 (NumValPoints))
    ALLOCATE (DistImage6B (NumValPoints))

     scx6 = 0
     icx6 = 0
     DistImage6B = 0.0

    call nn (b, NumValPoints, DrawSize, ValVector, m(:,:,d), scx6, DistImage6B)
```

```
call Recode (NumValPoints, scx6, SampleInfoClasses (:,d), icx6)

call ComputeAccuracyOnly (NumValPoints, icx6, NumValPoints, ValData, &
      ErrorMatrix6b(d,:,:), PercentAccuracy6b(d), AAPoints6b(:,:,d))


        do t = 1, NumValPoints

                B6SpecAAPoints (t,1,d) = scx6 (t)
                B6SpecDistAA (t,1,d) = DistImage6B (t)

        end do

DEALLOCATE (scx6)
DEALLOCATE (icx6)
DEALLOCATE (DistImage6B)

end do Classification6BLoop


PRINT *, "Sorting accuracies from six band loop."


Sort6bClassificationLoop: do i = 1, NumDraws

SortAccuracy(1,i) = i
SortAccuracy(2,i) = ErrorMatrix6b (i,1,1) + ErrorMatrix6b (i,2,2)

end do Sort6bClassificationLoop


call QuickSort(SortAccuracy, 1, NumDraws)


 open (unit = 46,  file = OutputFileNameSortAcc, access = "sequential", &
     form = "formatted", action = "write", status = "replace")

 WriteSortFile: do i = 1, NumDraws

 write (unit = 46, fmt = "(I10,I10)") SortAccuracy (1,i), SortAccuracy (2,i)

    end do WriteSortFile

 close (unit = 46)

 PRINT *, "Beginning draw specific processing loop..."

 ForestMapMarginal = 0.0
 StandardError = 0.0
 PrecisionFAE = 0.0
 PercentAccuracy = 0.0
 TotalClassForest = 0
 TotalClassNonforest = 0
 ErrorMatrix = 0
 ForestAreaEstimate = 0.0
 PercentAccuracyTr = 0.0
 ErrorMatrixTr = 0
 ForestAreaEstimateTr = 0.0
 StandardErrorTr = 0.0
 SpecDistAA = 0.0
 SpecAAPoints = 0
 OneTenthCount = 0

 DrawSpecificProcessingLoop: do i = 0, NumDraws, 10

        GreaterThenOne: if (i .eq. 0 ) then

                        IEdit = i + 1

                        end if GreaterThenOne
```

```
          EqualOne: if (i .NE. 0) then

                    IEdit = i

                    end if EqualOne

          PRINT *, "Sort number = ", IEdit

          OneTenthCount = OneTenthCount + 1

         ALLOCATE (scx (k))
          scx = 0
         ALLOCATE (icx (k))
          icx = 0
          ALLOCATE (DistanceImage (k))
         DistanceImage = 0.0

     PRINT *, "Nearest neighbor computation, draw ", IEdit

     SampleNumber = SortAccuracy(1,IEdit)

     WImageDrawNum (OneTenthCount) = SampleNumber

     call nn (b, k, DrawSize, x, m(:,:,SampleNumber), scx, DistanceImage)

     PRINT *, "Recode, draw ", IEdit

     call Recode (k, scx, SampleInfoClasses (:,SampleNumber), icx)

     PRINT *, "Computing accuracy plus area estimate, draw ", IEdit

     CALL AccPlusAreaEst (k, icx, NumValPoints, ValData, Acres,&
          ErrorMatrix(OneTenthCount,:,:), ForestAreaEstimate(OneTenthCount), &
          ForestMapMarginal(OneTenthCount), PercentAccuracy(OneTenthCount), &
          StandardError(OneTenthCount), PrecisionFAE(OneTenthCount), &
          AAPoints (:,:,OneTenthCount))

 TotalClassForest (OneTenthCount) = ErrorMatrix (OneTenthCount,1,2) + &
        ErrorMatrix (OneTenthCount,2,2)

 TotalClassNonforest (OneTenthCount) = ErrorMatrix (OneTenthCount,1,1) + &
         ErrorMatrix (OneTenthCount,2,1)

         do j = 1, NumValPoints

             SpecAAPoints (j,1,OneTenthCount) = scx (ValData (j,2))
                  SpecDistAA (j,1,OneTenthCount) = DistanceImage (ValData (j,2))
                  NumTrPointW (j,1,OneTenthCount) = &
                       RandomIndices (scx (ValData (j,2)),SampleNumber)
                  TrValXdiff = XYValData (j,1) - &
                       SampleXYTrain (1,scx(ValData(j,2)),SampleNumber)
                  TrValYdiff = XYValData (j,2) - &
                       SampleXYTrain (2,scx(ValData(j,2)),SampleNumber)
                  TrValXYDistW = SQRT ((TrValXdiff*TrValXdiff) + &
        (TrValYdiff*TrValYdiff))
`                  SpatialDistW (j,1,OneTenthCount) = TrValXYDistW
        end do

     PRINT *, "Computing McRoberts's forest area estimate, draw ", IEdit

CALL ComputeMcRobertsAreaEstimate (k, icx, NumValPoints, ValData, &
 ForestMapMarginal(OneTenthCount), PlotProportionForest, Acres, &
 TotalClassForest(OneTenthCount), TotalClassNonforest(OneTenthCount), &
 MeanPPPForest(OneTenthCount), MeanPPNonforest(OneTenthCount), &
 WeightedProportion (OneTenthCount), ForestArea (OneTenthCount), &
 ForestStratVar(OneTenthCount), NonforestStratVar(OneTenthCount), &
 VarianceSum (OneTenthCount), VarianceForestArea (OneTenthCount), &
 FAEPrecision (OneTenthCount))

     PRINT *, "Computing accuracy plus area estimate train, draw ", IEdit
```

```
        CALL AccPlusAreaEstP2 (k, icx, NumTrainingPoints, TrainValData, &
           ForestMapMarginal(OneTenthCount), ErrorMatrixTr(OneTenthCount,:,:), &
            ForestAreaEstimateTr(OneTenthCount), PercentAccuracyTr(OneTenthCount), &
            StandardErrorTr(OneTenthCount), AAPointsTr (:,:,OneTenthCount))

       DEALLOCATE (scx)
        DEALLOCATE (icx)
        DEALLOCATE (DistanceImage)

    end do DrawSpecificProcessingLoop


    open (unit = 4, file = OutputFileNameKSAll, access = "sequential", &
           form = "formatted", action = "write", status = "replace")

    open (unit = 16,  file = OutputFileNameAAPoints, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 19,  file = OutputFileNameWAATrain, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 26,  file = OutputFileNameAreaEst2, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteFilesB: do i = 1, OneTenthNumDraws

        WriteAAPoints: do j = 1, NumValPoints

            write (unit = 16,  fmt = "(I10,I12,I10,I10,F12.3,I10,F15.5)") i, &
                  AAPoints(j,1,i), AAPoints(j,2,i), SpecAAPoints (j,1,i), &
                  SpecDistAA (j,1,i), NumTrPointW(j,1,i), SpatialDistW(j,1,i)
`
        end do WriteAAPoints

        write (unit = 4, fmt = &
           "(I10,I10,I10,I10,I10,F12.8,F12.8,F12.6,ES20.9,ES20.9,I10,I10)") &
            WImageDrawNum(i), ErrorMatrix(i,1,1), ErrorMatrix(i,1,2), &
                  ErrorMatrix(i,2,1), ErrorMatrix(i,2,2), ForestAreaEstimate(i), &
                  ForestMapMarginal(i), PercentAccuracy(i), StandardError(i), &
                  PrecisionFAE(i), TotalClassForest (i), TotalClassNonforest(i)

        write (unit = 19, fmt = "(I10,I10,I10,I10,F12.8,F12.8,F12.6,ES20.9)") &
            ErrorMatrixTr(i,1,1), ErrorMatrixTr(i,1,2), ErrorMatrixTr(i,2,1), &
            ErrorMatrixTr(i,2,2), ForestAreaEstimateTr(i), ForestMapMarginal(i), &
            PercentAccuracyTr(i), StandardErrorTr(i)

        write (unit = 26, fmt = &
           "(F12.8,F12.8,F12.8,F15.5,F12.8,F12.8,ES20.9,F20.5,ES20.9,I10,I10)")&
            MeanPPForest(i), MeanPPNonforest(i), WeightedProportion(i), &
                  ForestArea(i), ForestStratVar(i), NonforestStratVar(i), &
                  VarianceSum(i), VarianceForestArea(i), FAEPrecision(i), &
                  TotalClassForest(i), TotalClassNonforest(i)

       end do WriteFilesB

close (unit = 4)
close (unit = 16)
close (unit = 19)
close (unit = 26)

    ALLOCATE (TestAAPoints (DrawSize, 2, NumDraws))

    TestErrorMatrix = 0
    TestForestAreaEstimate = 0.0
    TestPercentAccuracy = 0.0
    TestForestMapMarginal = 0.0
    TestStandardError = 0.0
    SpecDistTest = 0.0
    TestPrecisionFAE = 0.0

    TestDrawProcessingLoop: do i = 1, NumDraws
```

```
           ALLOCATE (Testscx (DrawSize))
            Testscx = 0
           ALLOCATE (Testicx (DrawSize))
            Testicx = 0
             ALLOCATE (TestDistImage (DrawSize))
            TestDistImage = 0.0

         PRINT *, "Nearest neighbor test computation, draw ", i

         call nn (b, DrawSize, DrawSize, TestVector(:,:,i), m(:,:,i), Testscx, &
            TestDistImage)

         PRINT *, "Recode test, draw ", i

         call Recode (DrawSize, Testscx, SampleInfoClasses (:, i), Testicx)

         PRINT *, "Computing accuracy plus area estimate test, draw ", i

         CALL AccPlusAreaEst (DrawSize, Testicx, DrawSize, TestValData (:,:,i), &
             Acres, TestErrorMatrix(i,:,:), TestForestAreaEstimate(i), &
             TestForestMapMarginal(i), TestPercentAccuracy(i), TestStandardError(i), &
             TestPrecisionFAE(i), TestAAPoints (:,:,i))

            do j = 1, DrawSize

            TestXDiff = 0.0
            TestYDiff = 0.0

                     TestIndex = DrawSize + j
                     NumTrPtTest (j,1,i) = RandomIndices (TestIndex,i)
                     SpecClassTest (j,1,i) = Testscx (j)
                     SpecDistTest (j,1,i) = TestDistImage (j)
                     TestSpecIndex = DrawSize + Testscx(j)
                     NumTrPtSpecTest (j,1,i) = RandomIndices (TestSpecIndex, i)
                     TestXDiff = TestXYTrain (1,j,i) - SampleXYTrain (1,Testscx(j),i)
                     TestYDiff = TestXYTrain (2,j,i) - SampleXYTrain (2,Testscx(j),i)
                     SpatialDTest (j,1,i) = SQRT((TestXDiff*TestXDiff) + &
                           (TestYDiff*TestYDiff))

            end do

          DEALLOCATE (Testscx)
           DEALLOCATE (Testicx)
           DEALLOCATE (TestDistImage)

    end do TestDrawProcessingLoop

DEALLOCATE (TestValData)

    PRINT *, "Writing test files..."

    open (unit = 24,  file = OutputFileNameTestVectors, access = "sequential", &
     form = "formatted", action = "write", status = "replace")

    open (unit = 23,  file = OutputFileNameAATest, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 25,  file = OutputFileNameTestDist, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteTestFiles: do i = 1, NumDraws

       WriteTestVectors: do j = 1, DrawSize

           write (unit = 24,  fmt = "(I10, I10, I10, I10, I10, I10)") &
                  TestVector (:,j,i)

                       end do WriteTestVectors
        write (unit = 24, fmt = "(I10, I10, I10, I10, I10, I10)") &
                  -1, -1, -1, -1, -1,-1
```

```
        write (unit = 23, fmt ="(I10,I10,I10,I10,F12.8,F12.8,F12.6,ES20.9,ES20.9)")&
    TestErrorMatrix(i,1,1), TestErrorMatrix(i,1,2), TestErrorMatrix(i,2,1), &
         TestErrorMatrix(i,2,2), TestForestAreaEstimate(i), &
           TestForestMapMarginal(i), TestPercentAccuracy(i), TestStandardError(i), &
           TestPrecisionFAE(i)

        WriteSpecTestDist: do j = 1, DrawSize

        write (unit = 25, fmt = "(I10,I10,I10,I10,I10,F11.3,I10,F15.5)") i, &
           TestAAPoints (j,1,i), TestAAPoints (j,2,i), NumTrPtTest(j,1,i), &
           SpecClassTest (j,1,i), SpecDistTest (j,1,i), NumTrPtSpecTest(j,1,i), &
           SpatialDTest(j,1,i)

                        end do WriteSpecTestDist

   end do WriteTestFiles

close (unit = 24)
close (unit = 23)
close (unit = 25)

DEALLOCATE (TestAAPoints)

    PRINT *, "Train processing loop..."

    TrainPercentAccuracy = 0.0
    TrainForestMapMarginal = 0.0
    TrainStandardError = 0.0
    TrainForestAreaEstimate = 0.0
    SpatialDistTr = 0.0
    SpecDistTrain = 0.0
    TrainPrecisionFAE = 0.0

    TrainDrawProcessingLoop: do i = 1, NumDraws

           ALLOCATE (Trainscx (NumTrainingPoints))
            Trainscx = 0
           ALLOCATE (Trainicx (NumTrainingPoints))
            Trainicx = 0
            ALLOCATE (TrainDistImage (NumTrainingPoints))
            TrainDistImage = 0.0

        PRINT *, "Nearest neighbor train computation, draw ", i

        call nn (b, NumTrainingPoints, DrawSize, TrainVector, m(:,:,i), Trainscx,&
           TrainDistImage)

        PRINT *, "Recode train, draw ", i

        call Recode (NumTrainingPoints, Trainscx, SampleInfoClasses(:,i), Trainicx)

        PRINT *, "Computing accuracy plus area estimate train, draw ", i

        CALL AccPlusAreaEst (NumTrainingPoints, Trainicx, NumTrainingPoints, &
             TrainValDataB, Acres, TrainErrorMatrix(i,:,:), &
               TrainForestAreaEstimate(i), TrainForestMapMarginal(i), &
               TrainPercentAccuracy(i), TrainStandardError(i), &
             TrainPrecisionFAE(i), TrainAAPoints (:,:,i))

            do j = 1, NumTrainingPoints

            TrainXDiff = 0.0
            TrainYDiff = 0.0

                    SpecClassTrain (j,1,i) = Trainscx (j)
                    SpecDistTrain (j,1,i) = TrainDistImage (j)
                    NumTrPtTrain (j,1,i) = RandomIndices (Trainscx(j),i)
                    TrainXDiff = XYTrain (j,1) - SampleXYTrain (1,Trainscx(j),i)
                    TrainYDiff = XYTrain (j,2) - SampleXYTrain (2,Trainscx(j),i)
                    SpatialDistTr (j,1,i) = SQRT((TrainXDiff*TrainXDiff) + &
```

```
                        (TrainYDiff*TrainYDiff))
           end do

          DEALLOCATE (Trainscx)
          DEALLOCATE (Trainicx)
          DEALLOCATE (TrainDistImage)

     end do TrainDrawProcessingLoop


     PRINT *, "Writing mean vectors..."

     open (unit = 15,  file = OutputFileNameMeanVectors, access = "sequential", &
      form = "formatted", action = "write", status = "replace")

     WriteMeanFiles: do i = 1, NumDraws

             MeanVectors: do j = 1, DrawSize

              write (unit = 15,  fmt = "(F10.1, F10.1, F10.1, F10.1, F10.1, F10.1)") &
                     m (:,j,i)

         end do MeanVectors

         write (unit = 15, fmt = "(I10, I10, I10, I10, I10, I10)") &
              -1, -1, -1, -1, -1,-1

        end do WriteMeanFiles

close (unit = 15)


     PRINT *, "Writing train files..."

     open (unit = 44,  file = OutputFileNameAATrain, access = "sequential", &
         form = "formatted", action = "write", status = "replace")

     open (unit = 18,  file = OutputFileNameTrainDistA, access = "sequential", &
         form = "formatted", action = "write", status = "replace")

     open (unit = 51,  file = OutputFileNameTrainDistB, access = "sequential", &
         form = "formatted", action = "write", status = "replace")

     WriteTrainFiles: do i = 1, NumDraws

         write (unit = 44, fmt ="(I10,I10,I10,I10,F12.8,F12.8,F12.6,ES20.9,ES20.9)")&
            TrainErrorMatrix(i,1,1), TrainErrorMatrix(i,1,2), &
            TrainErrorMatrix(i,2,1), TrainErrorMatrix(i,2,2), &
            TrainForestAreaEstimate(i), TrainForestMapMarginal(i), &
            TrainPercentAccuracy(i), TrainStandardError(i), TrainPrecisionFAE(i)

    end do WriteTrainFiles

    HalfDraws = NumDraws/2
    PRINT *, "HalfDraws = ", HalfDraws

         WriteSpecTrDA: do i = 1, HalfDraws

         WriteSpecTrainDistA: do j = 1, NumTrainingPoints

         write (unit = 18, fmt = "(I10,I10,I10,I10,F11.3,I10,F15.5)") i, &
             TrainAAPoints (j,1,i), TrainAAPoints (j,2,i), SpecClassTrain (j,1,i), &
             SpecDistTrain (j,1,i), NumTrPtTrain (j,1,i), SpatialDistTr(j,1,i)

                  end do WriteSpecTrainDistA

         end do WriteSpecTrDA

         WriteSpecTrDB: do i = HalfDraws+1, NumDraws

         WriteSpecTrainDistB: do j = 1, NumTrainingPoints
```

```
        write (unit = 51, fmt = "(I10,I10,I10,I10,F11.3,I10,F15.5)") i, &
           TrainAAPoints (j,1,i), TrainAAPoints (j,2,i), SpecClassTrain (j,1,i), &
           SpecDistTrain (j,1,i), NumTrPtTrain (j,1,i), SpatialDistTr(j,1,i)

                 end do WriteSpecTrainDistB

        end do WriteSpecTrDB

close (unit = 44)
close (unit = 18)
close (unit = 51)


    PRINT *, "Calculating sample sizes by information class..."

    ForestSampleSize = 0
    NonforestSampleSize = 0

    ComputeSampleSizes: do i = 1, NumDraws

           do j = 1, DrawSize

                  if (SampleInfoClasses (j,i) == 1) then

                           NonforestSampleSize (i) = NonforestSampleSize (i) + 1

                  end if

                  if (SampleInfoClasses (j,i) == 2) then

                           ForestSampleSize (i) = ForestSampleSize (i) + 1

                  end if

            end do

end do ComputeSampleSizes

    PRINT *, "Writing values all file..."

    open (unit = 3, file = OutputFileNameValuesAll, access = "sequential", &
           form = "formatted", action = "write", status = "replace")

    WriteSFiles: do i = 1, NumDraws

        WriteSamples: do j = 1, DrawSize

            write (unit = 3, fmt = &
                  "(I10,I10,I5,F18.9,F18.9,I10,I10,I5,F18.9,F18.9)")  &
               RandomIndices (j,i), SampleLocations (j,i), &
                  SampleInfoClasses (j,i), SampleXYTrain (1,j,i), &
                  SampleXYTrain (2,j,i), TestRandomIndices (j,i), &
               TestLocations (j,i), TestInfoClasses (j,i), TestXYTrain (1,j,i), &
                  TestXYTrain (2,j,i)

        end do WriteSamples

        write (unit = 3, fmt = "(I10,I10,I10,I10,I10,I10,I10,I10)") &
           -1,-1,-1,-1,-1,-1,-1,-1,-1,-1

    end do WriteSFiles

close (unit = 3)


    PRINT *, "Writing 6 band combination files..."

    open (unit = 42,  file = OutputFileNameBandCombo6AA, access = "sequential", &
        form = "formatted", action = "write", status = "replace")
```

```
    open (unit = 43,  file = OutputFileNameB6AAPoints, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteB6Files: do i = 1, NumDraws

            write (unit = 42, fmt = "(I10,I10,I10,I10,I10,F12.6,I10,I10)") i, &
                  ErrorMatrix6b(i,1,1), ErrorMatrix6b(i,1,2), ErrorMatrix6b (i,2,1), &
                  ErrorMatrix6b (i,2,2), PercentAccuracy6b (i), &
                  NonforestSampleSize(i), ForestSampleSize(i)


        WriteB6ComboAAPoints: do j = 1, NumValPoints

            write (unit = 43,  fmt = "(I10,I12,I10,I10,F12.3)") i, &
                  AAPoints6b (j,1,i), AAPoints6b (j,2,i), &
               B6SpecAAPoints (j,1,i), B6SpecDistAA (j,1,i)

        end do WriteB6ComboAAPoints

       end do WriteB6Files

close (unit = 42)
close (unit = 43)


    PRINT *, "Calculating spatial distance between training points..."

    TrminusTr = 0.0

    FindClosestSpatialTr2Tr: do i = 1, NumDraws

    do j = 1, DrawSize

   MinTrXYDist = 1000000.0
    TrNearestTrain = 0

           do l = 1, DrawSize

            if (l .NE. j) then


                  if (RandomIndices(l,i) .NE. RandomIndices(j,i)) then

                      TrXdiffTr = XYTrain(RandomIndices (j,i),1) - &
                                XYTrain(RandomIndices (l,i),1)
                      TrYdiffTr = XYTrain(RandomIndices (j,i),2) - &
                                XYTrain(RandomIndices (l,i),2)
                      TrDistTr = SQRT((TrXdiffTr*TrXdiffTr)+(TrYdiffTr*TrYdiffTr))
                      if (TrDistTr .LT. MinTrXYDist) then

                         MinTrXYDist = TrDistTr
                            TrNearestTrain = RandomIndices(l,i)

                            end if

                  end if

            end if

            end do
                    TrMinXY2TrainA (j,1,i) = RandomIndices(j,i)
                    TrMinXY2Train (j,1,i) = MinTrXYDist
                    TrMinXY2TrainA (j,2,i) = TrNearestTrain

                          TrminusTr (:,1) = TrainVector(:,RandomIndices(j,i)) - &
                                TrainVector(:,TrNearestTrain)

                  OnebyOneMatrix = &
                                SQRT (MATMUL (TRANSPOSE (TrminusTr), &
                                   TrminusTr))
```

```
                    TempTrDTr = OnebyOneMatrix (1,1)
                    TrMinXY2Train (j,2,i) = TempTrDTr

            end do

    end do FindClosestSpatialTr2Tr


    PRINT *, "Writing spatial distance between training points file..."

    open (unit = 47,  file = OutputFileNameTrMinXYTr, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteTrSFiles: do i = 1, NumDraws

        WriteTrSpatialTr: do j = 1, DrawSize

            write (unit = 47, fmt = "(I10,I10,I10,F15.5,I10,F15.5)") i, j, &
            TrMinXY2TrainA(j,1,i), TrMinXY2Train(j,1,i), TrMinXY2TrainA(j,2,i),&
            TrMinXY2Train(j,2,i)

        end do WriteTrSpatialTr

    end do WriteTrSFiles

close (unit = 47)


    PRINT *, "Calculating spatial distance between training and validation..."

    ValminusTr = 0.0

    FindClosestSpatialTr2Val: do i = 1, NumDraws

            do j = 1, NumValPoints

                   MinXYDist = 1000000.0
                    NearestTrain = 0

                    do l = 1, DrawSize

                    TrainValXDiff = SampleXYTrain (1,l,i) - XYValData (j,1)
                    TrainValYDiff = SampleXYTrain (2,l,i) - XYValData (j,2)
                    TrainValXYDist = SQRT((TrainValXDiff*TrainValXDiff) + &
                          (TrainValYDiff*TrainValYDiff))

                    if (TrainValXYDist .LT. MinXYDist) then

                            MinXYDist = TrainValXYDist
                            NearestTrain = RandomIndices(l,i)

                    end if

                    end do

                    ValMinXY2Train (j,1,i) = MinXYDist
                    ValMinXY2TrainA (j,1,i) = NearestTrain

                          ValminusTr (:,1) = ValVector(:,j) - &
                                  TrainVector(:,NearestTrain)

                  OnebyOneMatrixB = &
                                  SQRT (MATMUL (TRANSPOSE (ValminusTr), &
                                     ValminusTr))

                    TempValDTr = OnebyOneMatrixB (1,1)

                  ValMinXY2Train (j,2,i) = TempValDTr

            end do
```

```
        end do FindClosestSpatialTr2Val


        PRINT *, "Writing spatial distance file between train and val..."

        open (unit = 48,  file = OutputFileNameValMinXYTr, access = "sequential", &
            form = "formatted", action = "write", status = "replace")

        WriteFiles: do i = 1, NumDraws

            WriteValSpatialTr: do j = 1, NumValPoints

                write (unit = 48, fmt = "(I10,I10,F15.5,I10,F15.5)") i, j, &
                ValMinXY2Train(j,1,i), ValMinXY2TrainA(j,1,i), ValMinXY2Train(j,2,i)

            end do WriteValSpatialTr

           end do WriteFiles

    close (unit = 48)


    write (unit = *, fmt = *) "Global Processing Completed"

    write (unit = *, fmt = *) "Remember that draws are separated by two -1s in value files."

end program ksbootfa
```

APPENDIX C: Fortran 90 Code for Simulations with Stratified Random Sampling

(Note: Array sizes set for image VA15)

```
! Last change:  RHW  17 Apr 2005    2:31 pm
!
! ksbootstrat
! Randolph H. Wynne, Department of Forestry, Virginia Polytechnic Institute and State
! University
! Christine E. Blinn, Department of Forestry, Virginia Polytechnic Institute and State
! University
!
! Version 0.1
!
! Program start date: July 29, 2003
! Inputs:
! Outputs:

program ksbootstrat

   USE CEARSwLib
   USE quicksortw_mod

   implicit none

        INTEGER, parameter :: S_LEN = 32
        INTEGER (KIND = 4) :: d, i, j, k, l, p, q, r, s, t, v, w, y, z
        INTEGER (KIND = 4) :: ForestIndex, NonforestIndex
        INTEGER (KIND = 4) :: BandsLessOne, BandsLessTwo, BandsLessThree, BandsLessFour
        INTEGER (KIND = 4) :: LPlusOne, LPlusOneB, PPlusOne
        INTEGER (KIND = 4) :: JPlusOne, JPlusOneB, JPlusOneC
        INTEGER (KIND = 4) :: IPlusOne, IPlusOneB, IPlusOneC, IPlusOneD
        INTEGER (KIND = 4) :: OneTenthCount, OneTenthNumDraws, SampleNumber, IEdit
        INTEGER (KIND = 4) :: OnePlusJ
        INTEGER (KIND = 4) :: DrawSize
        INTEGER (KIND = 4) :: NumDraws
        INTEGER (KIND = 4) :: NumValPoints, NumTrainingPoints
        INTEGER (KIND = 4) :: SeedSize
        INTEGER (KIND = 2) :: a, b, c, e, f, g
        INTEGER (KIND = 4) :: ImageNumNonzeroPixels
        INTEGER (KIND = 4) :: NearestTrain, TrNearestTrain
        INTEGER (KIND = 4) :: TestIndex, TestSpecIndex
        REAL (KIND = 8) :: Xdiff
        REAL (KIND = 8) :: Ydiff
        REAL (KIND = 8) :: TrValXdiff
        REAL (KIND = 8) :: TrValYdiff
        REAL (KIND = 8) :: TrValXYDistW
        REAL (KIND = 8) :: XTrValDiff
        REAL (KIND = 8) :: YTrValDiff
        REAL (KIND = 8) :: TestXDiff
        REAL (KIND = 8) :: TestYDiff
        REAL (KIND = 8) :: TrainXDiff
        REAL (KIND = 8) :: TrainYDiff
        REAL (KIND = 8) :: TrXdiffTr
        REAL (KIND = 8) :: TrYdiffTr
        REAL (KIND = 8) :: TrDistTr
        REAL (KIND = 8) :: Acres
        REAL (KIND = 8) :: MinXYDist, MinTrXYDist
        REAL (KIND = 8) :: TrainValXYDist
        REAL (KIND = 8) :: TrainValXDiff
        REAL (KIND = 8) :: TrainValYDiff
        REAL (KIND = 8) :: TempTrDTr
        REAL (KIND = 8) :: TempValDTr
        REAL (KIND = 8) :: OnebyOneMatrix (1,1)
        REAL (KIND = 8) :: OnebyOneMatrixB (1,1)

        INTEGER (KIND = 4) :: ForestSampleSize, NonforestSampleSize
        INTEGER (KIND = 4) :: ForestPoolSize, NonforestPoolSize
        REAL (KIND = 8) :: TempRandomNumber, SingleRandomNumber
```

```
REAL (KIND = 8) :: ForestProportion, NonforestProportion
REAL (KIND = 8) :: ForestPoolSizeReal, NonforestPoolSizeReal
REAL (KIND = 8) :: NumTrainingPointsReal

INTEGER (KIND = 2) :: InfoClasses (41702)
INTEGER (KIND = 4) :: Locations (41702)
REAL (KIND = 8) :: XYTrain (41702,2)
REAL (KIND = 8) :: XYValData (1742,2)
INTEGER (KIND = 2) :: x (6,80126720)
INTEGER (KIND = 4) :: ValData (1742,2)
REAL (KIND = 8) :: PlotProportionForest (1742)
INTEGER (KIND = 4) :: RandomIndices (500,100)
INTEGER (KIND = 4) :: TestRandomIndices (500,100)
REAL (KIND = 8) :: SampleXYTrain (2,500,100)
INTEGER (KIND = 4) :: SampleInfoClasses (500,100)
INTEGER (KIND = 4) :: SampleLocations (500,100)
INTEGER (KIND = 4) :: TestInfoClasses (500,100)
INTEGER (KIND = 4) :: TestLocations (500,100)
REAL (KIND = 8) :: TestXYTrain (2,500,100)
INTEGER (KIND = 4) :: TrainValData (41702,2)
INTEGER (KIND = 4) :: TrainValDataB (41702,2)
REAL (KIND = 8) :: m (6,500,100)
INTEGER (KIND = 2) :: TestVector (6,500,100)
INTEGER (KIND = 2) :: TrainVector (6,41702)
INTEGER (KIND = 2) :: ValVector (6,1742)
REAL (KIND = 8) :: m1 (1,500,100)
INTEGER (KIND = 2) :: x1 (1,1742)
INTEGER (KIND = 4) :: ErrorMatrix1b (6,100,2,2)
REAL (KIND = 8) :: PercentAccuracy1b (6,100)
INTEGER (KIND = 4) :: AAPoints1b (1742,2,6,100)
INTEGER (KIND = 4) :: B1SpecAAPoints (1742,1,6,100)
REAL (KIND = 8) :: B1SpecDistAA (1742,1,6,100)
REAL (KIND = 8) :: m2 (2,500,100)
INTEGER (KIND = 2) :: x2 (2,1742)
INTEGER (KIND = 4) :: ErrorMatrix2b (15,100,2,2)
REAL (KIND = 8) :: PercentAccuracy2b (15,100)
INTEGER (KIND = 4) :: AAPoints2b (1742,2,15,100)
INTEGER (KIND = 4) :: B2SpecAAPoints (1742,1,15,100)
REAL (KIND = 8) :: B2SpecDistAA (1742,1,15,100)
REAL (KIND = 8) :: m3 (3,500,100)
INTEGER (KIND = 2) :: x3 (3,1742)
INTEGER (KIND = 4) :: ErrorMatrix3b (20,100,2,2)
REAL (KIND = 8) :: PercentAccuracy3b (20,100)
INTEGER (KIND = 4) :: AAPoints3b (1742,2,20,100)
INTEGER (KIND = 4) :: B3SpecAAPoints (1742,1,20,100)
REAL (KIND = 8) :: B3SpecDistAA (1742,1,20,100)
REAL (KIND = 8) :: m4 (4,500,100)
INTEGER (KIND = 2) :: x4 (4,1742)
INTEGER (KIND = 4) :: ErrorMatrix4b (15,100,2,2)
REAL (KIND = 8) :: PercentAccuracy4b (15,100)
INTEGER (KIND = 4) :: AAPoints4b (1742,2,15,100)
INTEGER (KIND = 4) :: B4SpecAAPoints (1742,1,15,100)
REAL (KIND = 8) :: B4SpecDistAA (1742,1,15,100)
REAL (KIND = 8) :: m5 (5,500,100)
INTEGER (KIND = 2) :: x5 (5,1742)
INTEGER (KIND = 4) :: ErrorMatrix5b (6,100,2,2)
REAL (KIND = 8) :: PercentAccuracy5b (6,100)
INTEGER (KIND = 4) :: AAPoints5b (1742,2,6,100)
INTEGER (KIND = 4) :: B5SpecAAPoints (1742,1,6,100)
REAL (KIND = 8) :: B5SpecDistAA (1742,1,6,100)
INTEGER (KIND = 4) :: ErrorMatrix6b (100,2,2)
REAL (KIND = 8) :: PercentAccuracy6b (100)
INTEGER (KIND = 4) :: AAPoints6b (1742,2,100)
INTEGER (KIND = 4) :: B6SpecAAPoints (1742,1,100)
REAL (KIND = 8) :: B6SpecDistAA (1742,1,100)
INTEGER (KIND = 4) :: SortAccuracy (2,100)
INTEGER (KIND = 4) :: WImageDrawNum (11)
REAL (KIND = 8) :: ForestMapMarginal (11)
REAL (KIND = 8) :: StandardError (11)
INTEGER (KIND = 4) :: AAPoints (1742,2,11)
REAL (KIND = 8) :: PrecisionFAE (11)
```

```
 INTEGER (KIND = 4) :: TotalClassForest (11)
INTEGER (KIND = 4) :: TotalClassNonforest (11)
REAL (KIND = 8) :: MeanPPForest (11)
REAL (KIND = 8) :: MeanPPNonforest (11)
REAL (KIND = 8) :: WeightedProportion (11)
REAL (KIND = 8) :: ForestArea (11)
REAL (KIND = 8) :: ForestStratVar (11)
REAL (KIND = 8) :: NonforestStratVar (11)
REAL (KIND = 8) :: VarianceSum (11)
REAL (KIND = 8) :: VarianceForestArea (11)
REAL (KIND = 8) :: FAEPrecision (11)
REAL (KIND = 8) :: PercentAccuracy (11)
INTEGER (KIND = 4) :: ErrorMatrix (11,2,2)
REAL (KIND = 8) :: ForestAreaEstimate (11)
REAL (KIND = 8) :: PercentAccuracyTr (11)
INTEGER (KIND = 4) :: ErrorMatrixTr (11,2,2)
REAL (KIND = 8) :: ForestAreaEstimateTr (11)
REAL (KIND = 8) :: StandardErrorTr (11)
INTEGER (KIND = 4) :: AAPointsTr (41702,2,11)
INTEGER (KIND = 4) :: SpecAAPoints (1742,1,11)
REAL (KIND = 8) :: SpecDistAA (1742,1,11)
INTEGER (KIND = 4) :: NumTrPointW (1742,1,11)
REAL (KIND = 8) :: SpatialDistW (1742,1,11)
INTEGER (KIND = 4) :: TestErrorMatrix (100,2,2)
REAL (KIND = 8) :: TestForestAreaEstimate (100)
REAL (KIND = 8) :: TestPercentAccuracy (100)
REAL (KIND = 8) :: TestForestMapMarginal (100)
REAL (KIND = 8) :: TestStandardError (100)
REAL (KIND = 8) :: SpecDistTest (500,1,100)
REAL (KIND = 8) :: TestPrecisionFAE (100)
INTEGER (KIND = 4) :: NumTrPtTest (500,1,100)
INTEGER (KIND = 4) :: NumTrPtSpecTest (500,1,100)
REAL (KIND = 8) :: SpatialDTest (500,1,100)
INTEGER (KIND = 4) :: SpecClassTest (500,1,100)
REAL (KIND = 8) :: TrainPercentAccuracy (100)
REAL (KIND = 8) :: TrainForestMapMarginal (100)
REAL (KIND = 8) :: TrainStandardError (100)
INTEGER (KIND = 4) :: TrainAAPoints (41702,2,100)
INTEGER (KIND = 4) :: TrainErrorMatrix (100,2,2)
REAL (KIND = 8) :: TrainForestAreaEstimate (100)
REAL (KIND = 8) :: SpatialDistTr (41702,1,100)
INTEGER (KIND = 4) :: SpecClassTrain (41702,1,100)
INTEGER (KIND = 4) :: NumTrPtTrain (41702,1,100)
REAL (KIND = 8) :: SpecDistTrain (41702,1,100)
REAL (KIND = 8) :: TrainPrecisionFAE (100)
INTEGER (KIND = 4) :: TrMinXY2TrainA (500,2,100)
REAL (KIND = 8) :: TrMinXY2Train (500,2,100)
REAL (KIND = 8) :: TrminusTr (6,1)
INTEGER (KIND = 4) :: ValMinXY2TrainA (1742,1,100)
REAL (KIND = 8) :: ValMinXY2Train (1742,2,100)
REAL (KIND = 8) :: ValminusTr (6,1)


INTEGER (KIND = 4), ALLOCATABLE :: TestValData (:,:,:)
INTEGER (KIND = 4), ALLOCATABLE :: TestAAPoints (:,:,:)
INTEGER (KIND = 2), ALLOCATABLE :: scx (:), icx (:)
INTEGER (KIND = 2), ALLOCATABLE :: Testscx (:), Testicx (:)
INTEGER (KIND = 2), ALLOCATABLE :: Trainscx (:), Trainicx (:)

REAL (KIND = 8), ALLOCATABLE :: DistanceImage (:)
REAL (KIND = 8), ALLOCATABLE :: TestDistImage (:)
REAL (KIND = 8), ALLOCATABLE :: TrainDistImage (:)

INTEGER (KIND = 2), ALLOCATABLE :: scx1 (:), icx1 (:)
INTEGER (KIND = 2), ALLOCATABLE :: scx2 (:), icx2 (:)
INTEGER (KIND = 2), ALLOCATABLE :: scx3 (:), icx3 (:)
INTEGER (KIND = 2), ALLOCATABLE :: scx4 (:), icx4 (:)
INTEGER (KIND = 2), ALLOCATABLE :: scx5 (:), icx5 (:)
INTEGER (KIND = 2), ALLOCATABLE :: scx6 (:), icx6 (:)
REAL (KIND = 8), ALLOCATABLE :: DistImage1B (:)
REAL (KIND = 8), ALLOCATABLE :: DistImage2B (:)
```

```
 REAL (KIND = 8), ALLOCATABLE :: DistImage3B (:)
 REAL (KIND = 8), ALLOCATABLE :: DistImage4B (:)
 REAL (KIND = 8), ALLOCATABLE :: DistImage5B (:)
 REAL (KIND = 8), ALLOCATABLE :: DistImage6B (:)

 INTEGER (KIND = 4), ALLOCATABLE :: ForestRandomIndices (:,:)
 INTEGER (KIND = 4), ALLOCATABLE :: NonforestRandomIndices (:,:)
 INTEGER (KIND = 4), ALLOCATABLE :: ForestLocations (:)
 INTEGER (KIND = 4), ALLOCATABLE :: NonforestLocations (:)
 INTEGER (KIND = 4), ALLOCATABLE :: ForestSampleLocations (:,:)
 INTEGER (KIND = 4), ALLOCATABLE :: NonforestSampleLocations (:,:)
 INTEGER (KIND = 4), ALLOCATABLE :: ForestTestLocations (:,:)
 INTEGER (KIND = 4), ALLOCATABLE :: NonforestTestLocations (:,:)
 INTEGER (KIND = 4), ALLOCATABLE :: TestFRandomIndices (:,:)
 INTEGER (KIND = 4), ALLOCATABLE :: TestNfRandomIndices (:,:)

 REAL (KIND = 8), ALLOCATABLE :: ForestXYTrain (:,:)
 REAL (KIND = 8), ALLOCATABLE :: NonforestXYTrain (:,:)
 REAL (KIND = 8), ALLOCATABLE :: ForestSampleXYTrain (:,:,:)
 REAL (KIND = 8), ALLOCATABLE :: NonforestSampleXYTrain (:,:,:)
 REAL (KIND = 8), ALLOCATABLE :: ForestTestXYTrain (:,:,:)
 REAL (KIND = 8), ALLOCATABLE :: NonforestTestXYTrain (:,:,:)
 REAL (KIND = 8), ALLOCATABLE :: ForestRandomNumbers (:,:)
 REAL (KIND = 8), ALLOCATABLE :: NonforestRandomNumbers (:,:)
 REAL (KIND = 8), ALLOCATABLE :: TestFRandomNumbers (:,:)
 REAL (KIND = 8), ALLOCATABLE :: TestNfRandomNumbers (:,:)

 CHARACTER (LEN = S_LEN) :: InputFileNameImage
 CHARACTER (LEN = S_LEN) :: InputFileNameTrainingData
 CHARACTER (LEN = S_LEN) :: InputFileNameTrainingXY
 CHARACTER (LEN = S_LEN) :: InputFileNameValidationData
 CHARACTER (LEN = S_LEN) :: InputFileNamePlotProportion
 CHARACTER (LEN = S_LEN) :: InputFileNameValDataXY
 CHARACTER (LEN = S_LEN) :: OutputFileNameRoot
 CHARACTER (LEN = S_LEN) :: OutputFileNameValuesAll
 CHARACTER (LEN = S_LEN) :: OutputFileNameKSAll
 CHARACTER (LEN = S_LEN) :: OutputFileNameMeanVectors
 CHARACTER (LEN = S_LEN) :: OutputFileNameAAPoints
 CHARACTER (LEN = S_LEN) :: OutputFileNameXYDistance
 CHARACTER (LEN = S_LEN) :: OutputFileNameTrainDistA
 CHARACTER (LEN = S_LEN) :: OutputFileNameWAATrain
 CHARACTER (LEN = S_LEN) :: OutputFileNameAATrain
 CHARACTER (LEN = S_LEN) :: OutputFileNameTestDist
 CHARACTER (LEN = S_LEN) :: OutputFileNameAATest
 CHARACTER (LEN = S_LEN) :: OutputFileNameTestVectors
 CHARACTER (LEN = S_LEN) :: OutputFileNameAreaEst2
 CHARACTER (LEN = S_LEN) :: OutputFileNameTrValXYDist
 CHARACTER (LEN = S_LEN) :: OutputFileNameSpecDistTrain
 CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo1AA
 CHARACTER (LEN = S_LEN) :: OutputFileNameB1AAPoints
 CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo2AA
 CHARACTER (LEN = S_LEN) :: OutputFileNameB2AAPoints
 CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo3AA
 CHARACTER (LEN = S_LEN) :: OutputFileNameB3AAPoints
 CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo4AA
 CHARACTER (LEN = S_LEN) :: OutputFileNameB4AAPoints
 CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo5AA
 CHARACTER (LEN = S_LEN) :: OutputFileNameB5AAPoints
 CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo6AA
 CHARACTER (LEN = S_LEN) :: OutputFileNameB6AAPoints
 CHARACTER (LEN = S_LEN) :: OutputFileNameSpecDValTr
 CHARACTER (LEN = S_LEN) :: OutputFileNameSortAcc
 CHARACTER (LEN = S_LEN) :: OutputFileNameTrMinXYTr
 CHARACTER (LEN = S_LEN) :: OutputFileNameValMinXYTr
 CHARACTER (LEN = S_LEN) :: OutputFileNameValVectors
 CHARACTER (LEN = S_LEN) :: OutputFileNameTrainVectors

open (10, file='strat_15.in', status = "old", action = "read")

read (10, *) DrawSize
 PRINT *, "Draw Size: ", DrawSize
```

```
read (10, *) NumDraws
 PRINT *, "Number of Draws: ", NumDraws

read (10, *) InputFileNameImage
 PRINT *, "Input File Name Raw Image: ", InputFileNameImage

 read (10, *) k
 PRINT *, "Number of Pixels in Raw Image: ", k

 read (10, *) b
 PRINT *, "Number of Bands: ", b

read (10, *) ImageNumNonzeroPixels
PRINT *, "Number of Nonzero Pixels in Raw Image: ", ImageNumNonzeroPixels

 read (10, *) InputFileNameTrainingData
 PRINT *, "Input File Name Training Data: ", InputFileNameTrainingData

read (10, *) InputFileNameTrainingXY
 PRINT *, "Input File Name Training XY: ", InputFileNameTrainingXY

 read (10, *) NumTrainingPoints
 PRINT *, "Number of Training Points: ", NumTrainingPoints

 read (10, *) InputFileNameValidationData
 PRINT *, "Input File Name Validation Data: ", InputFileNameValidationData

read (10, *) InputFileNameValDataXY
PRINT *, "Input File Name Validation XY: ", InputFileNameValDataXY

read (10, *) InputFileNamePlotProportion
PRINT *, "Input File Name Plot Proportion: ", InputFileNamePlotProportion

 read (10, *) NumValPoints
 PRINT *, "Number of Validation Points: ",  NumValPoints

 read (10, *) OutputFileNameRoot
PRINT *, "Output File Name Root (no suffix, e.g. 1000x1000): ", OutputFileNameRoot


 OutputFileNameValuesAll = TRIM (OutputFileNameRoot) // "Values_All.txt"
 OutputFileNameKSAll = TRIM (OutputFileNameRoot) // "KS_All.txt"
 OutputFileNameMeanVectors = TRIM (OutputFileNameRoot) // "Mean_Vectors.txt"
 OutputFileNameAAPoints = TRIM (OutputFileNameRoot) // "AAPoints.txt"
 OutputFileNameXYDistance = TRIM (OutputFileNameRoot) // "XYDistance.txt"
 OutputFileNameTrainDistA = TRIM (OutputFileNameRoot) // "TrainDistA.txt"
 OutputFileNameWAATrain = TRIM (OutputFileNameRoot) // "WAATrain.txt"
 OutputFileNameAATrain = TRIM (OutputFileNameRoot) // "AATrain.txt"
 OutputFileNameTestDist = TRIM (OutputFileNameRoot) // "TestDist.txt"
 OutputFileNameAATest = TRIM (OutputFileNameRoot) // "AATest.txt"
 OutputFileNameTestVectors = TRIM (OutputFileNameRoot) // "Test_Vectors.txt"
 OutputFileNameAreaEst2 = TRIM (OutputFileNameRoot) // "AreaEst2.txt"
 OutputFileNameTrValXYDist = TRIM (OutputFileNameRoot) // "TrValXYDist.txt"
 OutputFileNameSpecDistTrain = TRIM (OutputFileNameRoot) // "SpecDistTrain.txt"
 OutputFileNameBandCombo1AA = TRIM (OutputFileNameRoot) // "BandCombo1AA.txt"
 OutputFileNameB1AAPoints = TRIM (OutputFileNameRoot) // "B1AAPoints.txt"
 OutputFileNameBandCombo2AA = TRIM (OutputFileNameRoot) // "BandCombo2AA.txt"
 OutputFileNameB2AAPoints = TRIM (OutputFileNameRoot) // "B2AAPoints.txt"
 OutputFileNameBandCombo3AA = TRIM (OutputFileNameRoot) // "BandCombo3AA.txt"
 OutputFileNameB3AAPoints = TRIM (OutputFileNameRoot) // "B3AAPoints.txt"
 OutputFileNameBandCombo4AA = TRIM (OutputFileNameRoot) // "BandCombo4AA.txt"
 OutputFileNameB4AAPoints = TRIM (OutputFileNameRoot) // "B4AAPoints.txt"
 OutputFileNameBandCombo5AA = TRIM (OutputFileNameRoot) // "BandCombo5AA.txt"
 OutputFileNameB5AAPoints = TRIM (OutputFileNameRoot) // "B5AAPoints.txt"
 OutputFileNameBandCombo6AA = TRIM (OutputFileNameRoot) // "BandCombo6AA.txt"
 OutputFileNameB6AAPoints = TRIM (OutputFileNameRoot) // "B6AAPoints.txt"
 OutputFileNameSpecDValTr = TRIM (OutputFileNameRoot) // "SpecDValTr.txt"
 OutputFileNameSortAcc = TRIM (OutputFileNameRoot) // "SortAcc.txt"
 OutputFileNameTrMinXYTr = TRIM (OutputFileNameRoot) // "TrMinXYTr.txt"
 OutputFileNameValMinXYTr = TRIM (OutputFileNameRoot) // "ValMinXYTr.txt"
```

```
OutputFileNameValVectors = TRIM (OutputFileNameRoot) // "Val_Vectors.txt"
OutputFileNameTrainVectors = TRIM (OutputFileNameRoot) // "Train_Vectors.txt"

write (unit = *, FMT = *) "Input file names: "
write (unit = *, FMT = *) "    ", InputFileNameImage
write (unit = *, FMT = *) "    ", InputFileNameTrainingData
write (unit = *, FMT = *) "    ", InputFileNameValidationData
write (unit = *, FMT = *) "    ", InputFileNamePlotProportion
write (unit = *, FMT = *) "    ", InputFileNameTrainingXY
write (unit = *, FMT = *) "    ", InputFileNameValDataXY

write (unit = *, FMT = *) "Draw Size: "
write (unit = *, FMT = *) "    ", DrawSize

write (unit = *, FMT = *) "Number of Draws: "
write (unit = *, FMT = *) "    ", NumDraws

write (unit = *, FMT = *) "Number of Bands: "
write (unit = *, FMT = *) "    ", b, "(Raw Image)"

write (unit = *, FMT = *) "Number of Pixels in raw image: "
write (unit = *, FMT = *) "    ", k, " (Total)"

write (unit = *, FMT = *) "    ", ImageNumNonzeroPixels, " (Nonzero)"

write (unit = *, FMT = *) "Number of Validation Points: "
write (unit = *, FMT = *) "    ", NumValPoints

write (unit = *, FMT = *) "Output file names: "
write (unit = *, FMT = *) "    ", OutputFileNameValuesAll
write (unit = *, FMT = *) "    ", OutputFileNameKSAll
write (unit = *, FMT = *) "    ", OutputFileNameMeanVectors
write (unit = *, FMT = *) "    ", OutputFileNameAAPoints
write (unit = *, FMT = *) "    ", OutputFileNameXYDistance
write (unit = *, FMT = *) "    ", OutputFileNameTrainDistA
write (unit = *, FMT = *) "    ", OutputFileNameWAATrain
write (unit = *, FMT = *) "    ", OutputFileNameAATrain
write (unit = *, FMT = *) "    ", OutputFileNameTestDist
write (unit = *, FMT = *) "    ", OutputFileNameAATest
write (unit = *, FMT = *) "    ", OutputFileNameTestVectors
write (unit = *, FMT = *) "    ", OutputFileNameAreaEst2
write (unit = *, FMT = *) "    ", OutputFileNameTrValXYDist
write (unit = *, FMT = *) "    ", OutputFileNameSpecDistTrain
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo1AA
write (unit = *, FMT = *) "    ", OutputFileNameB1AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo2AA
write (unit = *, FMT = *) "    ", OutputFileNameB2AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo3AA
write (unit = *, FMT = *) "    ", OutputFileNameB3AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo4AA
write (unit = *, FMT = *) "    ", OutputFileNameB4AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo5AA
write (unit = *, FMT = *) "    ", OutputFileNameB5AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo6AA
write (unit = *, FMT = *) "    ", OutputFileNameB6AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameSpecDValTr
write (unit = *, FMT = *) "    ", OutputFileNameSortAcc
write (unit = *, FMT = *) "    ", OutputFileNameTrMinXYTr
write (unit = *, FMT = *) "    ", OutputFileNameValMinXYTr
write (unit = *, FMT = *) "    ", OutputFileNameValVectors
write (unit = *, FMT = *) "    ", OutputFileNameTrainVectors

r = 2 * DrawSize
s = DrawSize + 1
OneTenthNumDraws = ((NumDraws/10) + 1)

 PRINT *, "OneTenthNumDraws = ", OneTenthNumDraws

TempTrDTr = 0.0
TempValDTr = 0.0
TestIndex = 0
```

```
TestSpecIndex = 0
Acres = 0.0
Xdiff = 0.0
Ydiff = 0.0
 TrainValXDiff = 0.0
TrainValYDiff = 0.0
TrValXdiff = 0.0
TrValYdiff = 0.0
TrValXYDistW = 0.0
TrXdiffTr = 0.0
TrYdiffTr = 0.0
TrainXDiff = 0.0
TrainYDiff = 0.0
TestXDiff = 0.0
TestYDiff = 0.0
XTrValDiff = 0.0
YTrValDiff = 0.0
BandsLessOne = 0
BandsLessTwo = 0
BandsLessThree = 0
BandsLessFour = 0
IPlusOne = 0
IPlusOneB = 0
IPlusOneC = 0
IPlusOneD = 0
JPlusOne = 0
JPlusOneB = 0
JPlusOneC = 0
LPlusOne = 0
LPlusOneB = 0
PPlusOne = 0
OnePlusJ = 0
NearestTrain = 0
NonforestStratVar = 0.0
ForestStratVar = 0.0
MeanPPForest = 0.0
MeanPPNonforest = 0.0

TestRandomIndices = 0
RandomIndices = 0
SampleLocations = 0
SampleInfoClasses = 0
PercentAccuracy = 0.0
ErrorMatrix = 0
ForestAreaEstimate = 0.0
ForestMapMarginal = 0.0
StandardError = 0.0
scx = 0
icx = 0
ForestSampleSize = 0
NonforestSampleSize = 0


open (unit = 1, file = InputFileNameTrainingData, access = "sequential", &
       form = "formatted")

do i = 1, NumTrainingPoints

        read (unit = 1, fmt = "(I1)") InfoClasses(i)

end do

PRINT *, "Information classes read."

do i = 1, NumTrainingPoints

        read (unit = 1, fmt = "(I10.1)") Locations (i)

end do
```

```
close (unit = 1)

PRINT *, "Locations read."


open (unit = 22, file = InputFileNameTrainingXY, access = "sequential", &
       form = "formatted")

do i = 1, NumTrainingPoints

        read (unit = 22, fmt = "(F18.9)" ) XYTrain (i,1)

end do

do i = 1, NumTrainingPoints

        read (unit = 22, fmt = "(F18.9)" ) XYTrain (i,2)

end do

close (unit = 22)

PRINT *, "X and Y of training points read."


open (unit = 31, file = InputFileNameValDataXY, access = "sequential", &
       form = "formatted")

do i = 1, NumValPoints

        read (unit = 31, fmt = "(F18.9)" ) XYValData (i,1)

end do

do i = 1, NumValPoints

        read (unit = 31, fmt = "(F18.9)" ) XYValData (i,2)

end do

close (unit = 31)

PRINT *, "X and Y of validation points read."


open (unit = 11, file = InputFileNameImage, &
       form = "binary", status = "old", action = "read")

write (unit = *, fmt = *) "Reading input image into memory..."

ReadRawPixelsLoop: do i = 1, k

       ReadRawBandsLoop: do j = 1, b

               read (unit = 11) x (j,i)

    end do ReadRawBandsLoop

end do ReadRawPixelsLoop


close (unit = 11)

write (unit = *, fmt = *) "Raw image written into memory..."


open (unit = 12, file = InputFileNameValidationData, access = "sequential", &
       form = "formatted")

PRINT *, "Reading validation data..."
```

```
        ReadValidationLocationsLoop: do i = 1, NumValPoints

                read (unit = 12, fmt = *) ValData (i,1)

        end do ReadValidationLocationsLoop

        ReadValidationValuesLoop: do i = 1, NumValPoints

                read (unit = 12, fmt = *) ValData (i,2)

        end do ReadValidationValuesLoop


        close (unit = 12)

        PRINT *, "Reading plot proportions..."

        PlotProportionForest = 0.0

        open (unit = 30, file = InputFileNamePlotProportion, access = "sequential", &
               form = "formatted")

        ReadPlotProportionFLoop: do i = 1, NumValPoints

                read (unit = 30, fmt = "(F12.10)")  PlotProportionForest (i)

        end do ReadPlotProportionFLoop

        close (unit = 30)

        PRINT *, "Begin specific processing for forest and nonforest"

!       Draw stratified random sample from forest and nonforest classes after
!       initializing variables and determining size of each subpopulation

        NonforestPoolSize = 0
        ForestPoolSize = 0
        ForestIndex = 0
        NonforestIndex = 0

        do i = 1,NumTrainingPoints

               if (InfoClasses (i) == 1) then

                      NonforestPoolSize = NonforestPoolSize + 1

               end if

               if (InfoClasses (i) == 2) then

                      ForestPoolSize = ForestPoolSize + 1

               end if

        end do

        print *, "Nonforest Pool Size = ", NonforestPoolSize
        print *, "Forest Pool Size = ", ForestPoolSize
        print *, "Number of Training Points = ", NumTrainingPoints

        NonforestPoolSizeReal = NonforestPoolSize
        ForestPoolSizeReal = ForestPoolSize
        NumTrainingPointsReal = NumTrainingPoints

        NonforestProportion = (NonforestPoolSizeReal / NumTrainingPoints)
        ForestProportion = (ForestPoolSizeReal / NumTrainingPoints)

        NonforestSampleSize = int ((NonforestProportion * DrawSize) + 0.5, 4)
        ForestSampleSize = int ((ForestProportion * DrawSize) + 0.5, 4)

        print *, "Nonforest Proportion = ", NonforestProportion
```

```
print *, "Forest Proportion = ", ForestProportion
print *, "Draw Size = ", DrawSize

print *, "Nonforest Sample Size = ", NonforestSampleSize
print *, "Forest Sample Size = ", ForestSampleSize

if ((NonforestSampleSize + ForestSampleSize) .LT. DrawSize) then

    call random_seed ()
    CALL random_number (SingleRandomNumber)

    if (SingleRandomNumber .LT. 0.5) then

        NonforestSampleSize = NonforestSampleSize + 1
        else

            ForestSampleSize = ForestSampleSize + 1

    end if

end if


if ((NonforestSampleSize + ForestSampleSize) .GT. DrawSize) then

    call random_seed ()
    CALL random_number (SingleRandomNumber)

    if (SingleRandomNumber .LT. 0.5) then

        NonforestSampleSize = NonforestSampleSize - 1
        else

            ForestSampleSize = ForestSampleSize - 1

    end if

end if

ALLOCATE (NonforestRandomNumbers (NonforestSampleSize, NumDraws))
ALLOCATE (NonforestRandomIndices (NonforestSampleSize, NumDraws))
ALLOCATE (ForestRandomNumbers (ForestSampleSize, NumDraws))
ALLOCATE (ForestRandomIndices (ForestSampleSize, NumDraws))
ALLOCATE (NonforestLocations (NonforestPoolSize))
ALLOCATE (ForestLocations (ForestPoolSize))
ALLOCATE (ForestSampleLocations (ForestSampleSize, NumDraws))
ALLOCATE (NonforestSampleLocations (NonforestSampleSize, NumDraws))
ALLOCATE (ForestXYTrain (2,ForestPoolSize))
ALLOCATE (NonforestXYTrain (2,NonforestPoolSize))
ALLOCATE (ForestSampleXYTrain (2,ForestSampleSize, NumDraws))
ALLOCATE (NonforestSampleXYTrain (2,NonforestSampleSize, NumDraws))

ForestRandomIndices = 0
NonforestRandomIndices = 0
ForestLocations = 0
NonforestLocations = 0
ForestSampleLocations = 0
NonforestSampleLocations = 0
ForestXYTrain = 0.0
NonforestXYTrain = 0.0
ForestSampleXYTrain = 0.0
NonforestSampleXYTrain = 0.0
ForestRandomNumbers = 0.0
NonforestRandomNumbers = 0.0

do i = 1, NumTrainingPoints

       if (InfoClasses (i) == 1) then

                  NonforestIndex = NonforestIndex + 1
                NonforestLocations (NonforestIndex) = Locations (i)
```

```
                NonforestXYTrain (:,NonforestIndex) = XYTrain (i,:)

              end if

              if (InfoClasses (i) == 2) then

                ForestIndex = ForestIndex + 1
                ForestLocations (ForestIndex) = Locations (i)
                ForestXYTrain (:,ForestIndex) = XYTrain (i,:)

              end if

end do

 call random_seed ()

 CALL random_number (NonforestRandomNumbers)

 PRINT *, "First Nonforest Random Number is: ", NonforestRandomNumbers (1,1)

 NonforestRandomIndices = NonforestRandomNumbers * NonforestPoolSize + 1

 CALL random_number (ForestRandomNumbers)

 PRINT *, "First Forest Random Number is: ", ForestRandomNumbers (1,1)

 ForestRandomIndices = ForestRandomNumbers * ForestPoolSize + 1

    do i = 1, NumDraws

        do j = 1, NonforestSampleSize

                NonforestSampleLocations (j,i) = &
                      NonforestLocations (NonforestRandomIndices (j,i))
                NonforestSampleXYTrain (:,j,i) = &
                      NonforestXYTrain (:,NonforestRandomIndices (j,i))

           end do
end do

    do i = 1, NumDraws

        do j = 1, ForestSampleSize

                ForestSampleLocations (j,i) = &
                      ForestLocations (ForestRandomIndices (j,i))
                ForestSampleXYTrain (:,j,i) = &
                      ForestXYTrain (:,ForestRandomIndices (j,i))

        end do

SampleLocations (1:NonforestSampleSize,i) = NonforestSampleLocations (:,i)
SampleLocations (NonforestSampleSize+1:DrawSize,i) = ForestSampleLocations (:,i)
SampleInfoClasses (1:NonforestSampleSize,i) = 1
SampleInfoClasses (NonforestSampleSize+1:DrawSize,i) = 2
SampleXYTrain (:,1:NonforestSampleSize,i) = NonforestSampleXYTrain (:,:,i)
SampleXYTrain (:,NonforestSampleSize+1:DrawSize,i) = ForestSampleXYTrain (:,:,i)
RandomIndices (1:NonforestSampleSize,i) = NonforestRandomIndices (:,i)
RandomIndices (NonforestSampleSize+1:DrawSize,i) = ForestRandomIndices (:,i)


    end do

         DEALLOCATE (NonforestRandomNumbers)
         DEALLOCATE (NonforestRandomIndices)
         DEALLOCATE (ForestRandomNumbers)
         DEALLOCATE (ForestRandomIndices)
         DEALLOCATE (ForestSampleLocations)
         DEALLOCATE (NonforestSampleLocations)
        DEALLOCATE (ForestSampleXYTrain)
        DEALLOCATE (NonforestSampleXYTrain)
```

```
 PRINT *, "Creating Test samples..."


 ALLOCATE (TestNfRandomNumbers (NonforestSampleSize, NumDraws))
 ALLOCATE (TestNfRandomIndices (NonforestSampleSize, NumDraws))
 ALLOCATE (TestFRandomNumbers (ForestSampleSize, NumDraws))
 ALLOCATE (TestFRandomIndices (ForestSampleSize, NumDraws))
 ALLOCATE (ForestTestLocations (ForestSampleSize, NumDraws))
 ALLOCATE (NonforestTestLocations (NonforestSampleSize, NumDraws))
 ALLOCATE (ForestTestXYTrain (2,ForestSampleSize, NumDraws))
 ALLOCATE (NonforestTestXYTrain (2,NonforestSampleSize, NumDraws))

 TestNfRandomNumbers = 0.0
 TestNfRandomIndices = 0
 TestFRandomNumbers = 0.0
 TestFRandomIndices = 0
 ForestTestLocations = 0
 NonforestTestLocations = 0
 ForestTestXYTrain = 0.0
 NonforestTestXYTrain = 0.0


 CALL random_number (TestNfRandomNumbers)

 PRINT *, "First Nonforest Test Random Number is: ", TestNfRandomNumbers (1,1)

 TestNfRandomIndices = TestNfRandomNumbers * NonforestPoolSize + 1

 CALL random_number (TestFRandomNumbers)

 PRINT *, "First Forest Test Random Number is: ", TestFRandomNumbers (1,1)

 TestFRandomIndices = TestFRandomNumbers * ForestPoolSize + 1

     do i = 1, NumDraws

         do j = 1, NonforestSampleSize

                  NonforestTestLocations (j,i) = &
                        NonforestLocations (TestNfRandomIndices (j,i))
                  NonforestTestXYTrain (:,j,i) = &
                        NonforestXYTrain (:,TestNfRandomIndices (j,i))

            end do
 end do

     do i = 1, NumDraws

         do j = 1, ForestSampleSize

                  ForestTestLocations (j,i) = &
                        ForestLocations (TestFRandomIndices (j,i))
                  ForestTestXYTrain (:,j,i) = &
                        ForestXYTrain (:,TestFRandomIndices (j,i))

         end do

TestLocations (1:NonforestSampleSize,i) = NonforestTestLocations (:,i)
TestLocations (NonforestSampleSize+1:DrawSize,i) = ForestTestLocations (:,i)
TestInfoClasses (1:NonforestSampleSize,i) = 1
TestInfoClasses (NonforestSampleSize+1:DrawSize,i) = 2
TestXYTrain (:,1:NonforestSampleSize,i) = NonforestTestXYTrain (:,:,i)
TestXYTrain (:,NonforestSampleSize+1:DrawSize,i) = ForestTestXYTrain (:,:,i)
TestRandomIndices (1:NonforestSampleSize,i) = TestNfRandomIndices (:,i)
TestRandomIndices (NonforestSampleSize+1:DrawSize,i) = TestFRandomIndices (:,i)

end do


          DEALLOCATE (NonforestLocations)
```

```
          DEALLOCATE (ForestLocations)
          DEALLOCATE (ForestXYTrain)
          DEALLOCATE (NonforestXYTrain)
          DEALLOCATE (TestNfRandomNumbers)
          DEALLOCATE (TestNfRandomIndices)
          DEALLOCATE (TestFRandomNumbers)
          DEALLOCATE (TestFRandomIndices)
          DEALLOCATE (ForestTestLocations)
          DEALLOCATE (NonforestTestLocations)
          DEALLOCATE (ForestTestXYTrain)
          DEALLOCATE (NonforestTestXYTrain)


  PRINT *, "Creating test validation data..."

  ALLOCATE (TestValData (DrawSize,2,NumDraws))

  TestValDataLoop: do i = 1, NumDraws

          do j = 1, DrawSize

         TestValData (j,1,i) = TestInfoClasses (j,i)
         TestValData (j,2,i) = j

          end do

  end do TestValDataLoop

  PRINT *, "Creating train validation data..."

  TrainValDataLoop: do i = 1, NumTrainingPoints

         TrainValData (i,1) = InfoClasses (i)
         TrainValData (i,2) = Locations (i)

  end do TrainValDataLoop


  TrainValDataBLoop: do i = 1, NumTrainingPoints

         TrainValDataB (i,1) = InfoClasses (i)
         TrainValDataB (i,2) = i

  end do TrainValDataBLoop


PRINT *, "Creating mean vectors"

  OuterMeanVectorsLoop: do i = 1, NumDraws

      InnerMeanVectorsLoop: do j = 1, DrawSize

          m (:, j, i) = x (:,SampleLocations (j,i))

      end do InnerMeanVectorsLoop

  end do OuterMeanVectorsLoop

  PRINT *, "Creating test vectors..."

  OuterTestVectorsLoop: do i = 1, NumDraws

      InnerTestVectorsLoop: do j = 1, DrawSize

          TestVector (:, j, i) = x (:, TestLocations (j,i))

      end do InnerTestVectorsLoop

  end do OuterTestVectorsLoop

  PRINT *, "Creating train vectors..."
```

```
        TrainingVectorsLoop: do i = 1, NumTrainingPoints

                TrainVector (:, i) = x (:, Locations (i))

        end do TrainingVectorsLoop

        open (unit = 50, file = OutputFileNameTrainVectors, access = "sequential", &
            form = "formatted", action = "write", status = "replace")

           WriteTrainVectors: do j = 1, NumTrainingPoints

                write (unit = 50,  fmt = "(I10, I10, I10, I10, I10, I10)") &
                      TrainVector (:,j)

            end do WriteTrainVectors

        close (unit = 50)


        PRINT *, "Creating validation vector..."

        ValidationVectorsLoop: do i = 1, NumValPoints

                ValVector (:, i) = x (:, ValData (i,2))

        end do ValidationVectorsLoop

        open (unit = 49,  file = OutputFileNameValVectors, access = "sequential", &
            form = "formatted", action = "write", status = "replace")

           WriteValVectors: do j = 1, NumValPoints

                write (unit = 49,  fmt = "(I10, I10, I10, I10, I10, I10)")&
                      ValVector (:,j)

            end do WriteValVectors

        close (unit = 49)


        Acres = (0.22239 * ImageNumNonzeroPixels)

        PRINT *, "Acres = ", Acres
        PRINT *, "Beginning band combinations, 1 band loop..."

        PRINT *, "Beginning draw specific processing loop..."

! Exploring best subset of bands with all possible band combinations using just the
! validation data

       a = 1

        ErrorMatrix1b = 0
        PercentAccuracy1b = 0.0
        B1SpecAAPoints = 0
        B1SpecDistAA = 0.0
        m1 = 0.0
        x1 = 0

       OneBandLoop: do i = 1, b

                m1(1,:,:) = m(i,:,:)
                x1(1,:) = ValVector(i,:)


                        Classification1BLoop: do d = 1, NumDraws

                       ALLOCATE (scx1 (NumValPoints))
                       ALLOCATE (icx1 (NumValPoints))
                       ALLOCATE (DistImage1B (NumValPoints))
```

```
                    scx1 = 0
                    icx1 = 0
                     DistImage1B = 0.0

                      call nn (a, NumValPoints, DrawSize, x1, m1(:,:,d), scx1, &
                           DistImage1B)

                      call Recode (NumValPoints, scx1, SampleInfoClasses (:,d), icx1)

                      call ComputeAccuracyOnly (NumValPoints, icx1, NumValPoints, &
                           ValData, ErrorMatrix1b(i,d,:,:), PercentAccuracy1b(i,d), &
                           AAPoints1b(:,:,i,d))


                    do t = 1, NumValPoints

                     B1SpecAAPoints (t,1,i,d) = scx1 (t)
                            B1SpecDistAA (t,1,i,d) = DistImage1B (t)

                    end do

                    DEALLOCATE (scx1)
                    DEALLOCATE (icx1)
                    DEALLOCATE (DistImage1B)

                     end do Classification1BLoop

                     m1 = 0.0
                     x1 = 0

    end do OneBandLoop

    open (unit = 32,  file = OutputFileNameBandCombo1AA, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 33,  file = OutputFileNameB1AAPoints, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteB1Files: do i = 1, NumDraws

        WriteBandCombo1: do j = 1, 6

            write (unit = 32, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
           ErrorMatrix1b (j,i,1,1), ErrorMatrix1b (j,i,1,2), &
                  ErrorMatrix1b (j,i,2,1), ErrorMatrix1b (j,i,2,2), &
                  PercentAccuracy1b (j,i)

                  end do WriteBandCombo1

        WriteB1ComboAAPoints: do l = 1, 6

            do j = 1, NumValPoints

            write (unit = 33,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
            AAPoints1b (j,1,l,i), AAPoints1b (j,2,l,i), &
           B1SpecAAPoints (j,1,l,i), B1SpecDistAA (j,1,l,i)

            end do

        end do WriteB1ComboAAPoints

       end do WriteB1Files

close (unit = 32)
close (unit = 33)


    PRINT *, "Beginning band combinations, 2 band loop..."

    z = 0
```

```
c = 2

BandsLessOne = b - 1

ErrorMatrix2b = 0
PercentAccuracy2b = 0.0
B2SpecAAPoints = 0
B2SpecDistAA = 0.0
m2 = 0.0
x2 = 0

TwoBandLoop: do i = 1, BandsLessOne

                IPlusOne = i + 1

                Band2of2Loop: do j = IPlusOne, b

                       m2(1,:,:) = m(i,:,:)
                       m2(2,:,:) = m(j,:,:)
                      x2(1,:) = ValVector(i,:)
                      x2(2,:) = ValVector(j,:)

                z = z + 1

                Classification2BLoop: do d = 1, NumDraws


              ALLOCATE (scx2 (NumValPoints))
              ALLOCATE (icx2 (NumValPoints))
              ALLOCATE (DistImage2B (NumValPoints))

               scx2 = 0
               icx2 = 0
                DistImage2B = 0.0

                 call nn (c, NumValPoints, DrawSize, x2, m2(:,:,d), scx2, &
              DistImage2B)

                 call Recode (NumValPoints, scx2, SampleInfoClasses (:,d), icx2)

                 call ComputeAccuracyOnly (NumValPoints, icx2, NumValPoints, &
              ValData, ErrorMatrix2b(z,d,:,:), PercentAccuracy2b(z,d), &
       AAPoints2b(:,:,z,d))

               do t = 1, NumValPoints

                       B2SpecAAPoints (t,1,z,d) = scx2 (t)
                       B2SpecDistAA (t,1,z,d) = DistImage2B (t)

              end do

              DEALLOCATE (scx2)
              DEALLOCATE (icx2)
              DEALLOCATE (DistImage2B)

               end do Classification2BLoop

               m2 = 0.0
               x2 = 0

               end do Band2of2Loop

end do TwoBandLoop


open (unit = 34,  file = OutputFileNameBandCombo2AA, access = "sequential", &
    form = "formatted", action = "write", status = "replace")

open (unit = 35,  file = OutputFileNameB2AAPoints, access = "sequential", &
    form = "formatted", action = "write", status = "replace")
```

```
    WriteB2Files: do i = 1, NumDraws

        WriteBandCombo2: do j = 1, 15

            write (unit = 34, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
           ErrorMatrix2b (j,i,1,1), ErrorMatrix2b (j,i,1,2), &
                  ErrorMatrix2b (j,i,2,1), ErrorMatrix2b (j,i,2,2),&
                  PercentAccuracy2b (j,i)

                  end do WriteBandCombo2

        WriteB2ComboAAPoints: do l = 1, 15

            do j = 1, NumValPoints

            write (unit = 35,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
            AAPoints2b (j,1,l,i), AAPoints2b (j,2,l,i), &
           B2SpecAAPoints (j,1,l,i), B2SpecDistAA (j,1,l,i)

            end do

        end do WriteB2ComboAAPoints

       end do WriteB2Files

close (unit = 34)
close (unit = 35)


    PRINT *, "Beginning band combinations, 3 band loop..."

    y = 0
    e = 3

    BandsLessTwo = b - 2

    ErrorMatrix3b = 0
    PercentAccuracy3b = 0.0
    B3SpecAAPoints = 0
    B3SpecDistAA = 0.0
    m3 = 0.0
    x3 = 0

   ThreeBandLoop: do i = 1, BandsLessTwo

        IPlusOneB = i + 1

        Band2of3Loop: do j = IPlusOneB, BandsLessOne

            JPlusOne = j + 1

            Band3of3Loop: do l = JPlusOne, b

               m3(1,:,:) = m(i,:,:)
               m3(2,:,:) = m(j,:,:)
               m3(3,:,:) = m(l,:,:)
                x3(1,:) = ValVector(i,:)
                x3(2,:) = ValVector(j,:)
                x3(3,:) = ValVector(l,:)

                    y = y + 1

                    Classification3BLoop: do d = 1, NumDraws

                  ALLOCATE (scx3 (NumValPoints))
                  ALLOCATE (icx3 (NumValPoints))
                  ALLOCATE (DistImage3B (NumValPoints))

                   scx3 = 0
                   icx3 = 0
                    DistImage3B = 0.0
```

```
                    call nn (e, NumValPoints, DrawSize, x3, m3(:,:,d), scx3, &
                         DistImage3B)

                    call Recode (NumValPoints, scx3, SampleInfoClasses (:,d), icx3)

                    call ComputeAccuracyOnly (NumValPoints, icx3, NumValPoints, &
                          ValData, ErrorMatrix3b(y,d,:,:), PercentAccuracy3b(y,d), &
                          AAPoints3b(:,:,y,d))


                  do t = 1, NumValPoints

                          B3SpecAAPoints (t,1,y,d) = scx3 (t)
                          B3SpecDistAA (t,1,y,d) = DistImage3B (t)

                  end do

                  DEALLOCATE (scx3)
                  DEALLOCATE (icx3)
                  DEALLOCATE (DistImage3B)

                   end do Classification3BLoop

                   m3 = 0.0
                   x3 = 0

          end do Band3of3Loop

      end do Band2of3Loop

  end do ThreeBandLoop


  open (unit = 36,  file = OutputFileNameBandCombo3AA, access = "sequential", &
      form = "formatted", action = "write", status = "replace")

  open (unit = 37,  file = OutputFileNameB3AAPoints, access = "sequential", &
      form = "formatted", action = "write", status = "replace")

  WriteB3Files: do i = 1, NumDraws

      WriteBandCombo3: do j = 1, 20

          write (unit = 36, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
                 ErrorMatrix3b (j,i,1,1), ErrorMatrix3b (j,i,1,2), &
                 ErrorMatrix3b (j,i,2,1), ErrorMatrix3b (j,i,2,2),&
                 PercentAccuracy3b (j,i)

                end do WriteBandCombo3

      WriteB3ComboAAPoints: do l = 1, 20

          do j = 1, NumValPoints

          write (unit = 37,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
          AAPoints3b (j,1,l,i), AAPoints3b (j,2,l,i), &
         B3SpecAAPoints (j,1,l,i), B3SpecDistAA (j,1,l,i)

          end do

      end do WriteB3ComboAAPoints

     end do WriteB3Files

close (unit = 36)
close (unit = 37)


  PRINT *, "Beginning band combinations, 4 band loop..."
```

```
w = 0
f = 4

BandsLessThree = b - 3

ErrorMatrix4b = 0
PercentAccuracy4b = 0.0
B4SpecAAPoints = 0
B4SpecDistAA = 0.0
m4 = 0.0
x4 = 0

FourBandLoop: do i = 1, BandsLessThree

    IPlusOneC = i + 1

    Band2of4Loop: do j = IPlusOneC, BandsLessTwo

        JPlusOneB = j + 1

        Band3of4Loop: do l = JPlusOneB, BandsLessOne

            LPlusOne = l + 1

            Band4of4Loop: do p = LPlusOne, b

           m4(1,:,:) = m(i,:,:)
           m4(2,:,:) = m(j,:,:)
           m4(3,:,:) = m(l,:,:)
           m4(4,:,:) = m(p,:,:)
            x4(1,:) = ValVector(i,:)
            x4(2,:) = ValVector(j,:)
            x4(3,:) = ValVector(l,:)
            x4(4,:) = ValVector(p,:)

                w = w + 1

                Classification4BLoop: do d = 1, NumDraws

              ALLOCATE (scx4 (NumValPoints))
              ALLOCATE (icx4 (NumValPoints))
              ALLOCATE (DistImage4B (NumValPoints))

               scx4 = 0
                icx4 = 0
             DistImage4B = 0.0

                call nn (f, NumValPoints, DrawSize, x4, m4(:,:,d), scx4, &
                      DistImage4B)

                call Recode (NumValPoints, scx4, SampleInfoClasses (:,d), icx4)

                call ComputeAccuracyOnly (NumValPoints, icx4, NumValPoints, &
                      ValData, ErrorMatrix4b(w,d,:,:), PercentAccuracy4b(w,d), &
                      AAPoints4b(:,:,w,d))


               do t = 1, NumValPoints

                      B4SpecAAPoints (t,1,w,d) = scx4 (t)
                      B4SpecDistAA (t,1,w,d) = DistImage4B (t)

              end do

              DEALLOCATE (scx4)
              DEALLOCATE (icx4)
              DEALLOCATE (DistImage4B)

               end do Classification4BLoop

               m4 = 0.0
```

```
                    x4 = 0

                end do Band4of4Loop

            end do Band3of4Loop

        end do Band2of4Loop

    end do FourBandLoop


    open (unit = 38,  file = OutputFileNameBandCombo4AA, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 39,  file = OutputFileNameB4AAPoints, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteB4Files: do i = 1, NumDraws

        WriteBandCombo4: do j = 1, 15

            write (unit = 38, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
                  ErrorMatrix4b (j,i,1,1), ErrorMatrix4b (j,i,1,2), &
                  ErrorMatrix4b (j,i,2,1), ErrorMatrix4b (j,i,2,2),&
           PercentAccuracy4b (j,i)

                  end do WriteBandCombo4

        WriteB4ComboAAPoints: do l = 1, 15

            do j = 1, NumValPoints

            write (unit = 39,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
            AAPoints4b (j,1,l,i), AAPoints4b (j,2,l,i), &
           B4SpecAAPoints (j,1,l,i), B4SpecDistAA (j,1,l,i)

            end do

        end do WriteB4ComboAAPoints

       end do WriteB4Files

close (unit = 38)
close (unit = 39)


    PRINT *, "Beginning band combinations, 5 band loop..."

    v = 0
    g = 5

    BandsLessFour = b - 4

    ErrorMatrix5b = 0
    PercentAccuracy5b = 0.0
    B5SpecAAPoints = 0
    B5SpecDistAA = 0.0
    m5 = 0.0
    x5 = 0

   FiveBandLoop: do i = 1, BandsLessFour

        IPlusOneD = i + 1

        Band2of5Loop: do j = IPlusOneD, BandsLessThree

            JPlusOneC = j + 1

            Band3of5Loop: do l = JPlusOneC, BandsLessTwo

                LPlusOneB = l + 1
```

```
            Band4of5Loop: do p = LPlusOneB, BandsLessOne

                PPlusOne = p + 1

                Band5of5Loop: do q = PPlusOne, b

                   m5(1,:,:) = m(i,:,:)
                   m5(2,:,:) = m(j,:,:)
                   m5(3,:,:) = m(l,:,:)
                   m5(4,:,:) = m(p,:,:)
                   m5(5,:,:) = m(q,:,:)
                    x5(1,:) = ValVector(i,:)
                    x5(2,:) = ValVector(j,:)
                    x5(3,:) = ValVector(l,:)
                    x5(4,:) = ValVector(p,:)
                    x5(5,:) = ValVector(q,:)

                v = v + 1

                Classification5BLoop: do d = 1, NumDraws

              ALLOCATE (scx5 (NumValPoints))
              ALLOCATE (icx5 (NumValPoints))
              ALLOCATE (DistImage5B (NumValPoints))

               scx5 = 0
               icx5 = 0
                DistImage5B = 0.0

                 call nn (g, NumValPoints, DrawSize, x5, m5(:,:,d), scx5, &
              DistImage5B)

                 call Recode (NumValPoints, scx5, SampleInfoClasses (:,d), icx5)

                 call ComputeAccuracyOnly (NumValPoints, icx5, NumValPoints, &
                       ValData, ErrorMatrix5b(v,d,:,:), PercentAccuracy5b(v,d), &
                       AAPoints5b(:,:,v,d))


               do t = 1, NumValPoints

                       B5SpecAAPoints (t,1,v,d) = scx5 (t)
                       B5SpecDistAA (t,1,v,d) = DistImage5B (t)

              end do

              DEALLOCATE (scx5)
              DEALLOCATE (icx5)
              DEALLOCATE (DistImage5B)

                end do Classification5BLoop

                m5 = 0.0
                x5 = 0

                end do Band5of5Loop

            end do Band4of5Loop

        end do Band3of5Loop

    end do Band2of5Loop

end do FiveBandLoop


open (unit = 40,  file = OutputFileNameBandCombo5AA, access = "sequential", &
    form = "formatted", action = "write", status = "replace")

open (unit = 41,  file = OutputFileNameB5AAPoints, access = "sequential", &
```

```
         form = "formatted", action = "write", status = "replace")

   WriteB5Files: do i = 1, NumDraws

         WriteBandCombo5: do j = 1, 6

             write (unit = 40, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
                   ErrorMatrix5b (j,i,1,1), ErrorMatrix5b (j,i,1,2), &
                    ErrorMatrix5b (j,i,2,1), ErrorMatrix5b (j,i,2,2),&
             PercentAccuracy5b (j,i)

                   end do WriteBandCombo5

         WriteB5ComboAAPoints: do l = 1, 6

            do j = 1, NumValPoints

             write (unit = 41,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
             AAPoints5b (j,1,l,i), AAPoints5b (j,2,l,i), &
            B5SpecAAPoints (j,1,l,i), B5SpecDistAA (j,1,l,i)

             end do

         end do WriteB5ComboAAPoints

        end do WriteB5Files

close (unit = 40)
close (unit = 41)


    PRINT *, "Beginning band combinations, 6 band loop..."

   ErrorMatrix6b = 0
   PercentAccuracy6b = 0.0
   B6SpecAAPoints = 0
   B6SpecDistAA = 0.0

   Classification6BLoop: do d = 1, NumDraws


   ALLOCATE (scx6 (NumValPoints))
   ALLOCATE (icx6 (NumValPoints))
   ALLOCATE (DistImage6B (NumValPoints))

    scx6 = 0
    icx6 = 0
    DistImage6B = 0.0

   call nn (b, NumValPoints, DrawSize, ValVector, m(:,:,d), scx6, DistImage6B)

   call Recode (NumValPoints, scx6, SampleInfoClasses (:,d), icx6)

   call ComputeAccuracyOnly (NumValPoints, icx6, NumValPoints, ValData, &
         ErrorMatrix6b(d,:,:), PercentAccuracy6b(d), AAPoints6b(:,:,d))


            do t = 1, NumValPoints

                   B6SpecAAPoints (t,1,d) = scx6 (t)
                   B6SpecDistAA (t,1,d) = DistImage6B (t)

           end do

   DEALLOCATE (scx6)
   DEALLOCATE (icx6)
   DEALLOCATE (DistImage6B)

   end do Classification6BLoop
```

```
   PRINT *, "Sorting accuracies from six band loop."


   Sort6bClassificationLoop: do i = 1, NumDraws

   SortAccuracy(1,i) = i
   SortAccuracy(2,i) = ErrorMatrix6b (i,1,1) + ErrorMatrix6b (i,2,2)

   end do Sort6bClassificationLoop


   call QuickSort(SortAccuracy, 1, NumDraws)


    open (unit = 46,  file = OutputFileNameSortAcc, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteSortFile: do i = 1, NumDraws

    write (unit = 46, fmt = "(I10,I10)") SortAccuracy (1,i), SortAccuracy (2,i)

       end do WriteSortFile

close (unit = 46)

    PRINT *, "Beginning draw specific processing loop..."

    ForestMapMarginal = 0.0
    StandardError = 0.0
    PrecisionFAE = 0.0
    PercentAccuracy = 0.0
    TotalClassForest = 0
    TotalClassNonforest = 0
    ErrorMatrix = 0
    ForestAreaEstimate = 0.0
    PercentAccuracyTr = 0.0
    ErrorMatrixTr = 0
    ForestAreaEstimateTr = 0.0
    StandardErrorTr = 0.0
    SpecDistAA = 0.0
    SpecAAPoints = 0
    OneTenthCount = 0

    DrawSpecificProcessingLoop: do i = 0, NumDraws, 10

           GreaterThenOne: if (i .eq. 0 ) then

                        IEdit = i + 1

                        end if GreaterThenOne

            EqualOne: if (i .NE. 0) then

                      IEdit = i

                      end if EqualOne

            PRINT *, "Sort number = ", IEdit

            OneTenthCount = OneTenthCount + 1

           ALLOCATE (scx (k))
            scx = 0
           ALLOCATE (icx (k))
            icx = 0
            ALLOCATE (DistanceImage (k))
           DistanceImage = 0.0

        PRINT *, "Nearest neighbor computation, draw ", IEdit

        SampleNumber = SortAccuracy(1,IEdit)
```

```
    WImageDrawNum (OneTenthCount) = SampleNumber

    call nn (b, k, DrawSize, x, m(:,:,SampleNumber), scx, DistanceImage)

    PRINT *, "Recode, draw ", IEdit

    call Recode (k, scx, SampleInfoClasses (:,SampleNumber), icx)

    PRINT *, "Computing accuracy plus area estimate, draw ", IEdit

    CALL AccPlusAreaEst (k, icx, NumValPoints, ValData, Acres,&
         ErrorMatrix(OneTenthCount,:,:), ForestAreaEstimate(OneTenthCount), &
         ForestMapMarginal(OneTenthCount), PercentAccuracy(OneTenthCount), &
         StandardError(OneTenthCount), PrecisionFAE(OneTenthCount), &
         AAPoints (:,:,OneTenthCount))

TotalClassForest (OneTenthCount) = ErrorMatrix (OneTenthCount,1,2) + &
       ErrorMatrix (OneTenthCount,2,2)

TotalClassNonforest (OneTenthCount) = ErrorMatrix (OneTenthCount,1,1) + &
        ErrorMatrix (OneTenthCount,2,1)

        do j = 1, NumValPoints

            SpecAAPoints (j,1,OneTenthCount) = scx (ValData (j,2))
                 SpecDistAA (j,1,OneTenthCount) = DistanceImage (ValData (j,2))
                 NumTrPointW (j,1,OneTenthCount) = &
                             RandomIndices (scx (ValData (j,2)),SampleNumber)
                 TrValXdiff = XYValData (j,1) - &
                             SampleXYTrain (1,scx(ValData(j,2)),SampleNumber)
                 TrValYdiff = XYValData (j,2) - &
                             SampleXYTrain (2,scx(ValData(j,2)),SampleNumber)
                 TrValXYDistW = SQRT ((TrValXdiff*TrValXdiff) + &
              (TrValYdiff*TrValYdiff))
                 SpatialDistW (j,1,OneTenthCount) = TrValXYDistW
       end do

    PRINT *, "Computing McRoberts's forest area estimate, draw ", IEdit

CALL ComputeMcRobertsAreaEstimate (k, icx, NumValPoints, ValData, &
ForestMapMarginal(OneTenthCount), PlotProportionForest, Acres, &
TotalClassForest(OneTenthCount), TotalClassNonforest(OneTenthCount), &
MeanPPForest(OneTenthCount), MeanPPNonforest(OneTenthCount), &
WeightedProportion (OneTenthCount), ForestArea (OneTenthCount), &
ForestStratVar(OneTenthCount), NonforestStratVar(OneTenthCount), &
VarianceSum (OneTenthCount), VarianceForestArea (OneTenthCount), &
FAEPrecision (OneTenthCount))

    PRINT *, "Computing accuracy plus area estimate train, draw ", IEdit

    CALL AccPlusAreaEstP2 (k, icx, NumTrainingPoints, TrainValData, &
       ForestMapMarginal(OneTenthCount), ErrorMatrixTr(OneTenthCount,:,:), &
        ForestAreaEstimateTr(OneTenthCount), PercentAccuracyTr(OneTenthCount), &
        StandardErrorTr(OneTenthCount), AAPointsTr (:,:,OneTenthCount))

     DEALLOCATE (scx)
    DEALLOCATE (icx)
    DEALLOCATE (DistanceImage)

end do DrawSpecificProcessingLoop


open (unit = 4, file = OutputFileNameKSAll, access = "sequential", &
       form = "formatted", action = "write", status = "replace")

open (unit = 16,  file = OutputFileNameAAPoints, access = "sequential", &
    form = "formatted", action = "write", status = "replace")

open (unit = 19,  file = OutputFileNameWAATrain, access = "sequential", &
    form = "formatted", action = "write", status = "replace")
```

```
    open (unit = 26,  file = OutputFileNameAreaEst2, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteFilesB: do i = 1, OneTenthNumDraws

        WriteAAPoints: do j = 1, NumValPoints

            write (unit = 16,  fmt = "(I10,I12,I10,I10,F12.3,I10,F15.5)") i, &
                  AAPoints(j,1,i), AAPoints(j,2,i), SpecAAPoints (j,1,i), &
                  SpecDistAA (j,1,i), NumTrPointW(j,1,i), SpatialDistW(j,1,i)

        end do WriteAAPoints

        write (unit = 4, fmt = &
           "(I10,I10,I10,I10,I10,F12.8,F12.8,F12.6,ES20.9,ES20.9,I10,I10)") &
            WImageDrawNum(i), ErrorMatrix(i,1,1), ErrorMatrix(i,1,2), &
                  ErrorMatrix(i,2,1), ErrorMatrix(i,2,2), ForestAreaEstimate(i), &
                  ForestMapMarginal(i), PercentAccuracy(i), StandardError(i), &
                  PrecisionFAE(i), TotalClassForest (i), TotalClassNonforest(i)

        write (unit = 19, fmt = "(I10,I10,I10,I10,F12.8,F12.8,F12.6,ES20.9)") &
            ErrorMatrixTr(i,1,1), ErrorMatrixTr(i,1,2), ErrorMatrixTr(i,2,1), &
            ErrorMatrixTr(i,2,2), ForestAreaEstimateTr(i), ForestMapMarginal(i), &
            PercentAccuracyTr(i), StandardErrorTr(i)

        write (unit = 26, fmt = &
           "(F12.8,F12.8,F12.8,F15.5,F12.8,F12.8,ES20.9,F20.5,ES20.9,I10,I10)")&
            MeanPPForest(i), MeanPPNonforest(i), WeightedProportion(i), &
                  ForestArea(i), ForestStratVar(i), NonforestStratVar(i), &
                  VarianceSum(i), VarianceForestArea(i), FAEPrecision(i), &
                  TotalClassForest(i), TotalClassNonforest(i)

       end do WriteFilesB

close (unit = 4)
close (unit = 16)
close (unit = 19)
close (unit = 26)

    ALLOCATE (TestAAPoints (DrawSize, 2, NumDraws))

    TestErrorMatrix = 0
    TestForestAreaEstimate = 0.0
    TestPercentAccuracy = 0.0
    TestForestMapMarginal = 0.0
    TestStandardError = 0.0
    SpecDistTest = 0.0
    TestPrecisionFAE = 0.0

    TestDrawProcessingLoop: do i = 1, NumDraws

          ALLOCATE (Testscx (DrawSize))
           Testscx = 0
          ALLOCATE (Testicx (DrawSize))
           Testicx = 0
           ALLOCATE (TestDistImage (DrawSize))
           TestDistImage = 0.0

        PRINT *, "Nearest neighbor test computation, draw ", i

        call nn (b, DrawSize, DrawSize, TestVector(:,:,i), m(:,:,i), Testscx, &
           TestDistImage)

        PRINT *, "Recode test, draw ", i

        call Recode (DrawSize, Testscx, SampleInfoClasses (:, i), Testicx)

        PRINT *, "Computing accuracy plus area estimate test, draw ", i

        CALL AccPlusAreaEst (DrawSize, Testicx, DrawSize, TestValData (:,:,i), &
```

```
            Acres, TestErrorMatrix(i,:,:), TestForestAreaEstimate(i), &
            TestForestMapMarginal(i), TestPercentAccuracy(i), TestStandardError(i), &
            TestPrecisionFAE(i), TestAAPoints (:,:,i))

          do j = 1, DrawSize

          TestXDiff = 0.0
          TestYDiff = 0.0

                  NumTrPtTest (j,1,i) = TestRandomIndices (j,i)
                  SpecClassTest (j,1,i) = Testscx (j)
                  SpecDistTest (j,1,i) = TestDistImage (j)
                  NumTrPtSpecTest (j,1,i) = TestRandomIndices (Testscx(j), i)
                  TestXDiff = TestXYTrain (1,j,i) - SampleXYTrain (1,Testscx(j),i)
                  TestYDiff = TestXYTrain (2,j,i) - SampleXYTrain (2,Testscx(j),i)
                  SpatialDTest (j,1,i) = SQRT((TestXDiff*TestXDiff) + &
                        (TestYDiff*TestYDiff))

          end do

        DEALLOCATE (Testscx)
        DEALLOCATE (Testicx)
        DEALLOCATE (TestDistImage)

    end do TestDrawProcessingLoop

DEALLOCATE (TestValData)

    PRINT *, "Writing test files..."

    open (unit = 24,  file = OutputFileNameTestVectors, access = "sequential", &
     form = "formatted", action = "write", status = "replace")

    open (unit = 23,  file = OutputFileNameAATest, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 25,  file = OutputFileNameTestDist, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteTestFiles: do i = 1, NumDraws

       WriteTestVectors: do j = 1, DrawSize

           write (unit = 24,  fmt = "(I10, I10, I10, I10, I10, I10)") &
                  TestVector (:,j,i)

                      end do WriteTestVectors
        write (unit = 24, fmt = "(I10, I10, I10, I10, I10, I10)") &
                  -1, -1, -1, -1, -1,-1

        write (unit = 23, fmt ="(I10,I10,I10,I10,F12.8,F12.8,F12.6,ES20.9,ES20.9)")&
   TestErrorMatrix(i,1,1), TestErrorMatrix(i,1,2), TestErrorMatrix(i,2,1), &
         TestErrorMatrix(i,2,2), TestForestAreaEstimate(i), &
          TestForestMapMarginal(i), TestPercentAccuracy(i), TestStandardError(i), &
          TestPrecisionFAE(i)

        WriteSpecTestDist: do j = 1, DrawSize

        write (unit = 25, fmt = "(I10,I10,I10,I10,I10,F11.3,I10,F15.5)") i, &
          TestAAPoints (j,1,i), TestAAPoints (j,2,i), NumTrPtTest(j,1,i), &
          SpecClassTest (j,1,i), SpecDistTest (j,1,i), NumTrPtSpecTest(j,1,i), &
          SpatialDTest(j,1,i)

                      end do WriteSpecTestDist

   end do WriteTestFiles

close (unit = 24)
close (unit = 23)
close (unit = 25)
```

```
    DEALLOCATE (TestAAPoints)

        PRINT *, "Train processing loop..."

        TrainPercentAccuracy = 0.0
        TrainForestMapMarginal = 0.0
        TrainStandardError = 0.0
        TrainForestAreaEstimate = 0.0
        SpatialDistTr = 0.0
        SpecDistTrain = 0.0
        TrainPrecisionFAE = 0.0

        TrainDrawProcessingLoop: do i = 1, NumDraws

               ALLOCATE (Trainscx (NumTrainingPoints))
                Trainscx = 0
               ALLOCATE (Trainicx (NumTrainingPoints))
                Trainicx = 0
                 ALLOCATE (TrainDistImage (NumTrainingPoints))
                TrainDistImage = 0.0

            PRINT *, "Nearest neighbor train computation, draw ", i

            call nn (b, NumTrainingPoints, DrawSize, TrainVector, m(:,:,i), Trainscx, &
               TrainDistImage)

            PRINT *, "Recode train, draw ", i

            call Recode (NumTrainingPoints, Trainscx, SampleInfoClasses(:,i), Trainicx)

            PRINT *, "Computing accuracy plus area estimate train, draw ", i

            CALL AccPlusAreaEst (NumTrainingPoints, Trainicx, NumTrainingPoints, &
                 TrainValDataB, Acres, TrainErrorMatrix(i,:,:), &
                      TrainForestAreaEstimate(i), &
                 TrainForestMapMarginal(i), TrainPercentAccuracy(i), &
                      TrainStandardError(i), &
                 TrainPrecisionFAE(i), TrainAAPoints (:,:,i))

                do j = 1, NumTrainingPoints

                TrainXDiff = 0.0
                TrainYDiff = 0.0

                         SpecClassTrain (j,1,i) = Trainscx (j)
                         SpecDistTrain (j,1,i) = TrainDistImage (j)
                         NumTrPtTrain (j,1,i) = RandomIndices (Trainscx(j),i)
                         TrainXDiff = XYTrain (j,1) - SampleXYTrain (1,Trainscx(j),i)
                         TrainYDiff = XYTrain (j,2) - SampleXYTrain (2,Trainscx(j),i)
                         SpatialDistTr (j,1,i) = SQRT((TrainXDiff*TrainXDiff) +
(TrainYDiff*TrainYDiff))
                end do

               DEALLOCATE (Trainscx)
                DEALLOCATE (Trainicx)
                DEALLOCATE (TrainDistImage)

        end do TrainDrawProcessingLoop


        PRINT *, "Writing mean vectors..."

        open (unit = 15,  file = OutputFileNameMeanVectors, access = "sequential", &
         form = "formatted", action = "write", status = "replace")

        WriteMeanFiles: do i = 1, NumDraws

                MeanVectors: do j = 1, DrawSize

                 write (unit = 15,  fmt = "(F10.1, F10.1, F10.1, F10.1, F10.1, F10.1)") &
                       m (:,j,i)
```

```
            end do MeanVectors

            write (unit = 15, fmt = "(I10, I10, I10, I10, I10, I10)") &
               -1, -1, -1, -1, -1,-1

           end do WriteMeanFiles

    close (unit = 15)


        PRINT *, "Writing train files..."

        open (unit = 44,  file = OutputFileNameAATrain, access = "sequential", &
            form = "formatted", action = "write", status = "replace")

        open (unit = 18,  file = OutputFileNameTrainDistA, access = "sequential", &
            form = "formatted", action = "write", status = "replace")

!        open (unit = 51,  file = OutputFileNameTrainDistB, access = "sequential", &
!            form = "formatted", action = "write", status = "replace")

        WriteTrainFiles: do i = 1, NumDraws

            write (unit = 44, fmt ="(I10,I10,I10,I10,F12.8,F12.8,F12.6,ES20.9,ES20.9)")&
               TrainErrorMatrix(i,1,1), TrainErrorMatrix(i,1,2), &
               TrainErrorMatrix(i,2,1), TrainErrorMatrix(i,2,2), &
               TrainForestAreaEstimate(i), TrainForestMapMarginal(i), &
               TrainPercentAccuracy(i), TrainStandardError(i), TrainPrecisionFAE(i)

       end do WriteTrainFiles

            WriteSpecTrDA: do i = 1, NumDraws

            WriteSpecTrainDistA: do j = 1, NumTrainingPoints

            write (unit = 18, fmt = "(I10,I10,I10,I10,F11.3,I10,F15.5)") i, &
                TrainAAPoints (j,1,i), TrainAAPoints (j,2,i), SpecClassTrain (j,1,i), &
                SpecDistTrain (j,1,i), NumTrPtTrain (j,1,i), SpatialDistTr(j,1,i)

                    end do WriteSpecTrainDistA

            end do WriteSpecTrDA


    close (unit = 44)
    close (unit = 18)

        PRINT *, "Writing values all file..."

        open (unit = 3, file = OutputFileNameValuesAll, access = "sequential", &
               form = "formatted", action = "write", status = "replace")

        WriteSFiles: do i = 1, NumDraws

            WriteSamples: do j = 1, DrawSize

                write (unit = 3, fmt = &
                      "(I10,I10,I5,F18.9,F18.9,I10,I10,I5,F18.9,F18.9)")  &
                   RandomIndices (j,i), SampleLocations (j,i), &
                      SampleInfoClasses (j,i), SampleXYTrain (1,j,i), &
                      SampleXYTrain (2,j,i), TestRandomIndices (j,i), &
                   TestLocations (j,i), TestInfoClasses (j,i), TestXYTrain (1,j,i), &
                      TestXYTrain (2,j,i)

            end do WriteSamples

            write (unit = 3, fmt = "(I10,I10,I10,I10,I10,I10,I10,I10)") &
               -1,-1,-1,-1,-1,-1,-1,-1,-1,-1

        end do WriteSFiles
```

```
close (unit = 3)

    PRINT *, "Writing 6 band combination files..."

    open (unit = 42,  file = OutputFileNameBandCombo6AA, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 43,  file = OutputFileNameB6AAPoints, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteB6Files: do i = 1, NumDraws

            write (unit = 42, fmt = "(I10,I10,I10,I10,I10,F12.6,I10,I10)") i, &
                  ErrorMatrix6b(i,1,1), ErrorMatrix6b(i,1,2), ErrorMatrix6b (i,2,1), &
                  ErrorMatrix6b (i,2,2), PercentAccuracy6b (i), &
                  NonforestSampleSize(i), ForestSampleSize(i)


        WriteB6ComboAAPoints: do j = 1, NumValPoints

            write (unit = 43,  fmt = "(I10,I12,I10,I10,F12.3)") i, &
            AAPoints6b (j,1,i), AAPoints6b (j,2,i), &
      B6SpecAAPoints (j,1,i), B6SpecDistAA (j,1,i)

        end do WriteB6ComboAAPoints

       end do WriteB6Files

close (unit = 42)
close (unit = 43)

    PRINT *, "Calculating spatial distance between training points..."

    TrminusTr = 0.0

    FindClosestSpatialTr2Tr: do i = 1, NumDraws

    do j = 1, DrawSize

   MinTrXYDist = 1000000.0
    TrNearestTrain = 0

           do l = 1, DrawSize

            if (l .NE. j) then


                  if (RandomIndices(l,i) .NE. RandomIndices(j,i)) then

                      TrXdiffTr = XYTrain(RandomIndices (j,i),1) - &
                                  XYTrain(RandomIndices (l,i),1)
                      TrYdiffTr = XYTrain(RandomIndices (j,i),2) - &
                                  XYTrain(RandomIndices (l,i),2)
                      TrDistTr = SQRT((TrXdiffTr*TrXdiffTr)+(TrYdiffTr*TrYdiffTr))
                      if (TrDistTr .LT. MinTrXYDist) then

                         MinTrXYDist = TrDistTr
                            TrNearestTrain = RandomIndices(l,i)

                            end if

                  end if

            end if

            end do
                    TrMinXY2TrainA (j,1,i) = RandomIndices(j,i)
                    TrMinXY2Train (j,1,i) = MinTrXYDist
                    TrMinXY2TrainA (j,2,i) = TrNearestTrain
```

```
                        TrminusTr (:,1) = TrainVector(:,RandomIndices(j,i)) - &
                                TrainVector(:,TrNearestTrain)

                    OnebyOneMatrix = &
                                SQRT (MATMUL (TRANSPOSE (TrminusTr), &
                                   TrminusTr))

                     TempTrDTr = OnebyOneMatrix (1,1)
                     TrMinXY2Train (j,2,i) = TempTrDTr

            end do

    end do FindClosestSpatialTr2Tr


    PRINT *, "Writing spatial distance between training points file..."

    open (unit = 47,  file = OutputFileNameTrMinXYTr, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteTrSFiles: do i = 1, NumDraws

        WriteTrSpatialTr: do j = 1, DrawSize

            write (unit = 47, fmt = "(I10,I10,I10,F15.5,I10,F15.5)") i, j, &
            TrMinXY2TrainA(j,1,i), TrMinXY2Train(j,1,i), TrMinXY2TrainA(j,2,i),&
            TrMinXY2Train(j,2,i)

        end do WriteTrSpatialTr

    end do WriteTrSFiles

close (unit = 47)


    PRINT *, "Calculating spatial distance between training and validation..."

    ValminusTr = 0.0

    FindClosestSpatialTr2Val: do i = 1, NumDraws

            do j = 1, NumValPoints

                   MinXYDist = 1000000.0
                    NearestTrain = 0

                    do l = 1, DrawSize

                    TrainValXDiff = SampleXYTrain (1,l,i) - XYValData (j,1)
                    TrainValYDiff = SampleXYTrain (2,l,i) - XYValData (j,2)
                    TrainValXYDist = SQRT((TrainValXDiff*TrainValXDiff) + &
                          (TrainValYDiff*TrainValYDiff))

                    if (TrainValXYDist .LT. MinXYDist) then

                            MinXYDist = TrainValXYDist
                            NearestTrain = RandomIndices(l,i)

                    end if

                    end do

                    ValMinXY2Train (j,1,i) = MinXYDist
                    ValMinXY2TrainA (j,1,i) = NearestTrain

                          ValminusTr (:,1) = ValVector(:,j) - &
                                TrainVector(:,NearestTrain)

                   OnebyOneMatrixB = &
                                SQRT (MATMUL (TRANSPOSE (ValminusTr), &
                                   ValminusTr))
```

```
                        TempValDTr = OnebyOneMatrixB (1,1)

                       ValMinXY2Train (j,2,i) = TempValDTr

                end do

        end do FindClosestSpatialTr2Val


        PRINT *, "Writing spatial distance file between train and val..."

        open (unit = 48,  file = OutputFileNameValMinXYTr, access = "sequential", &
            form = "formatted", action = "write", status = "replace")

        WriteFiles: do i = 1, NumDraws

            WriteValSpatialTr: do j = 1, NumValPoints

                write (unit = 48, fmt = "(I10,I10,F15.5,I10,F15.5)") i, j, &
                ValMinXY2Train(j,1,i), ValMinXY2TrainA(j,1,i), ValMinXY2Train(j,2,i)

            end do WriteValSpatialTr

           end do WriteFiles

    close (unit = 48)


    write (unit = *, fmt = *) "Global Processing Completed"

    write (unit = *, fmt = *) "Remember that draws are separated by two -1s in value files."

end program ksbootstrat
```

APPENDIX D: Fortran 90 Code for Simulations with Systematic Sampling

(Note: Array sizes set for image VA15)

```
! Last change:  RHW  26 Mar 2005   10:40 pm
!
! ksbootsys
! Randolph H. Wynne, Department of Forestry, Virginia Polytechnic Institute and State
! University
! Christine E. Blinn, Department of Forestry, Virginia Polytechnic Institute and State
! University
!
! Version 0.1
!
! Program start date: July 29, 2003
! Inputs:

! Outputs:

program ksbootsys

        USE CEARSwLib
        USE quicksortw_mod

   implicit none


         INTEGER, parameter :: S_LEN = 32 !, THREE_LINES = 198
        INTEGER (KIND = 4) :: d, i, j, k, l, p, q, r, s, t, v, w, y, z  ! Loop counters
        INTEGER (KIND = 4) :: BandsLessOne, BandsLessTwo, BandsLessThree, BandsLessFour
        INTEGER (KIND = 4) :: LPlusOne, LPlusOneB, PPlusOne
        INTEGER (KIND = 4) :: JPlusOne, JPlusOneB, JPlusOneC
        INTEGER (KIND = 4) :: IPlusOne, IPlusOneB, IPlusOneC, IPlusOneD
        INTEGER (KIND = 4) :: OneTenthCount, OneTenthNumDraws, SampleNumber
        INTEGER (KIND = 4) :: NumberWhole
        INTEGER (KIND = 4) :: OnePlusJ
        INTEGER (KIND = 4) :: DrawSize
        INTEGER (KIND = 4) :: NumDraws
        INTEGER (KIND = 4) :: HalfDraws
        INTEGER (KIND = 4) :: NumValPoints, NumTrainingPoints
        INTEGER (KIND = 4) :: SeedSize
        INTEGER (KIND = 2) :: a, b, c, e, f, g
        INTEGER (KIND = 4) :: ImageNumNonzeroPixels
        INTEGER (KIND = 4) :: NearestTrain, TrNearestTrain
        INTEGER (KIND = 4) :: TestIndex, TestSpecIndex
        INTEGER (KIND = 4) :: Remainder, LeftOver
        REAL (KIND = 8) :: NumDrawsReal
        REAL (KIND = 8) :: DrawSkipNumber
        REAL (KIND = 8) :: Xdiff
        REAL (KIND = 8) :: Ydiff
        REAL (KIND = 8) :: TrValXdiff
        REAL (KIND = 8) :: TrValYdiff
        REAL (KIND = 8) :: TrValXYDistW
        REAL (KIND = 8) :: XTrValDiff
        REAL (KIND = 8) :: YTrValDiff
        REAL (KIND = 8) :: TestXDiff
        REAL (KIND = 8) :: TestYDiff
        REAL (KIND = 8) :: TrainXDiff
        REAL (KIND = 8) :: TrainYDiff
        REAL (KIND = 8) :: TrXdiffTr
        REAL (KIND = 8) :: TrYdiffTr
        REAL (KIND = 8) :: TrDistTr
        REAL (KIND = 8) :: Acres
        REAL (KIND = 8) :: MinXYDist, MinTrXYDist
        REAL (KIND = 8) :: TrainValXYDist
        REAL (KIND = 8) :: TrainValXDiff
        REAL (KIND = 8) :: TrainValYDiff
        REAL (KIND = 8) :: TempTrDTr
        REAL (KIND = 8) :: TempValDTr
```

```
REAL (KIND = 8) :: OnebyOneMatrix (1,1)
REAL (KIND = 8) :: OnebyOneMatrixB (1,1)
REAL (KIND = 8) :: SingleRandomNumber

INTEGER (KIND = 2) :: InfoClasses (46903)
INTEGER (KIND = 4) :: Locations (46903)
INTEGER (KIND = 4) :: IEdit (101)
REAL (KIND = 8) :: XYTrain (46903,2)
REAL (KIND = 8) :: XYValData (1742,2)
INTEGER (KIND = 2) :: x (6,80126720)
INTEGER (KIND = 4) :: ValData (1742,2)
REAL (KIND = 8) :: PlotProportionForest (1742)
REAL (KIND = 8) :: TestRandomNumbers (496)
INTEGER (KIND = 4) :: TestRandomIndices (500,496)
INTEGER (KIND = 4) :: RandomIndices (500,496)
INTEGER (KIND = 4) :: FirstRandomIndices (496)
REAL (KIND = 8) :: SampleXYTrain (2,500,496)
INTEGER (KIND = 4) :: SampleInfoClasses (500,496)
INTEGER (KIND = 4) :: SampleLocations (500,496)
INTEGER (KIND = 4) :: TestInfoClasses (500,496)
INTEGER (KIND = 4) :: TestLocations (500,496)
REAL (KIND = 8) :: TestXYTrain (2,500,496)
INTEGER (KIND = 4) :: TrainValData (46903,2)
INTEGER (KIND = 4) :: TrainValDataB (46903,2)
REAL (KIND = 8) :: m (6,500,496)
INTEGER (KIND = 2) :: TestVector (6,500,496)
INTEGER (KIND = 2) :: TrainVector (6,46903)
INTEGER (KIND = 2) :: ValVector (6,1742)
REAL (KIND = 8) :: m1 (1,500,496)
INTEGER (KIND = 2) :: x1 (1,1742)
INTEGER (KIND = 4) :: ErrorMatrix1b (6,496,2,2)
REAL (KIND = 8) :: PercentAccuracy1b (6,496)
INTEGER (KIND = 4) :: AAPoints1b (1742,2,6,496)
INTEGER (KIND = 4) :: B1SpecAAPoints (1742,1,6,496)
REAL (KIND = 8) :: B1SpecDistAA (1742,1,6,496)
REAL (KIND = 8) :: m2 (2,500,496)
INTEGER (KIND = 2) :: x2 (2,1742)
INTEGER (KIND = 4) :: ErrorMatrix2b (15,496,2,2)
REAL (KIND = 8) :: PercentAccuracy2b (15,496)
INTEGER (KIND = 4) :: AAPoints2b (1742,2,15,496)
INTEGER (KIND = 4) :: B2SpecAAPoints (1742,1,15,496)
REAL (KIND = 8) :: B2SpecDistAA (1742,1,15,496)
REAL (KIND = 8) :: m3 (3,500,496)
INTEGER (KIND = 2) :: x3 (3,1742)
INTEGER (KIND = 4) :: ErrorMatrix3b (20,496,2,2)
REAL (KIND = 8) :: PercentAccuracy3b (20,496)
INTEGER (KIND = 4) :: AAPoints3b (1742,2,20,496)
INTEGER (KIND = 4) :: B3SpecAAPoints (1742,1,20,496)
REAL (KIND = 8) :: B3SpecDistAA (1742,1,20,496)
REAL (KIND = 8) :: m4 (4,500,496)
INTEGER (KIND = 2) :: x4 (4,1742)
INTEGER (KIND = 4) :: ErrorMatrix4b (15,496,2,2)
REAL (KIND = 8) :: PercentAccuracy4b (15,496)
INTEGER (KIND = 4) :: AAPoints4b (1742,2,15,496)
INTEGER (KIND = 4) :: B4SpecAAPoints (1742,1,15,496)
REAL (KIND = 8) :: B4SpecDistAA (1742,1,15,496)
REAL (KIND = 8) :: m5 (5,500,496)
INTEGER (KIND = 2) :: x5 (5,1742)
INTEGER (KIND = 4) :: ErrorMatrix5b (6,496,2,2)
REAL (KIND = 8) :: PercentAccuracy5b (6,496)
INTEGER (KIND = 4) :: AAPoints5b (1742,2,6,496)
INTEGER (KIND = 4) :: B5SpecAAPoints (1742,1,6,496)
REAL (KIND = 8) :: B5SpecDistAA (1742,1,6,496)
INTEGER (KIND = 4) :: ErrorMatrix6b (496,2,2)
REAL (KIND = 8) :: PercentAccuracy6b (496)
INTEGER (KIND = 4) :: AAPoints6b (1742,2,496)
INTEGER (KIND = 4) :: B6SpecAAPoints (1742,1,496)
REAL (KIND = 8) :: B6SpecDistAA (1742,1,496)
INTEGER (KIND = 4) :: SortAccuracy (2,496)
INTEGER (KIND = 4) :: WImageDrawNum (101)
REAL (KIND = 8) :: ForestMapMarginal (101)
```

```
REAL (KIND = 8) :: StandardError (101)
INTEGER (KIND = 4) :: AAPoints (1742,2,101)
REAL (KIND = 8) :: PrecisionFAE (101)
 INTEGER (KIND = 4) :: TotalClassForest (101)
INTEGER (KIND = 4) :: TotalClassNonforest (101)
REAL (KIND = 8) :: MeanPPForest (101)
REAL (KIND = 8) :: MeanPPNonforest (101)
REAL (KIND = 8) :: WeightedProportion (101)
REAL (KIND = 8) :: ForestArea (101)
REAL (KIND = 8) :: ForestStratVar (101)
REAL (KIND = 8) :: NonforestStratVar (101)
REAL (KIND = 8) :: VarianceSum (101)
REAL (KIND = 8) :: VarianceForestArea (101)
REAL (KIND = 8) :: FAEPrecision (101)
REAL (KIND = 8) :: PercentAccuracy (101)
INTEGER (KIND = 4) :: ErrorMatrix (101,2,2)
REAL (KIND = 8) :: ForestAreaEstimate (101)
REAL (KIND = 8) :: PercentAccuracyTr (101)
INTEGER (KIND = 4) :: ErrorMatrixTr (101,2,2)
REAL (KIND = 8) :: ForestAreaEstimateTr (101)
REAL (KIND = 8) :: StandardErrorTr (101)
INTEGER (KIND = 4) :: AAPointsTr (46903,2,101)
INTEGER (KIND = 4) :: SpecAAPoints (1742,1,101)
REAL (KIND = 8) :: SpecDistAA (1742,1,101)
INTEGER (KIND = 4) :: NumTrPointW (1742,1,101)
REAL (KIND = 8) :: SpatialDistW (1742,1,101)
INTEGER (KIND = 4) :: TestErrorMatrix (496,2,2)
REAL (KIND = 8) :: TestForestAreaEstimate (496)
REAL (KIND = 8) :: TestPercentAccuracy (496)
REAL (KIND = 8) :: TestForestMapMarginal (496)
REAL (KIND = 8) :: TestStandardError (496)
REAL (KIND = 8) :: SpecDistTest (500,1,496)
REAL (KIND = 8) :: TestPrecisionFAE (496)
INTEGER (KIND = 4) :: NumTrPtTest (500,1,496)
INTEGER (KIND = 4) :: NumTrPtSpecTest (500,1,496)
REAL (KIND = 8) :: SpatialDTest (500,1,496)
INTEGER (KIND = 4) :: SpecClassTest (500,1,496)
REAL (KIND = 8) :: TrainPercentAccuracy (496)
REAL (KIND = 8) :: TrainForestMapMarginal (496)
REAL (KIND = 8) :: TrainStandardError (496)
INTEGER (KIND = 4) :: TrainAAPoints (46903,2,496)
INTEGER (KIND = 4) :: TrainErrorMatrix (496,2,2)
REAL (KIND = 8) :: TrainForestAreaEstimate (496)
REAL (KIND = 8) :: SpatialDistTr (46903,1,496)
INTEGER (KIND = 4) :: SpecClassTrain (46903,1,496)
INTEGER (KIND = 4) :: NumTrPtTrain (46903,1,496)
REAL (KIND = 8) :: SpecDistTrain (46903,1,496)
REAL (KIND = 8) :: TrainPrecisionFAE (496)
INTEGER (KIND = 4) :: ForestSampleSize (496)
INTEGER (KIND = 4) :: NonforestSampleSize (496)
INTEGER (KIND = 4) :: TrMinXY2TrainA (500,2,496)
REAL (KIND = 8) :: TrMinXY2Train (500,2,496)
REAL (KIND = 8) :: TrminusTr (6,1)
INTEGER (KIND = 4) :: ValMinXY2TrainA (1742,1,496)
REAL (KIND = 8) :: ValMinXY2Train (1742,2,496)
REAL (KIND = 8) :: ValminusTr (6,1)

INTEGER, ALLOCATABLE :: Seed (:)
INTEGER (KIND = 4), ALLOCATABLE :: TestValData (:,:,:)
INTEGER (KIND = 4), ALLOCATABLE :: TestAAPoints (:,:,:)
INTEGER (KIND = 2), ALLOCATABLE :: scx (:), icx (:)
INTEGER (KIND = 2), ALLOCATABLE :: Testscx (:), Testicx (:)
INTEGER (KIND = 2), ALLOCATABLE :: Trainscx (:), Trainicx (:)

REAL (KIND = 8), ALLOCATABLE :: DistanceImage (:)
REAL (KIND = 8), ALLOCATABLE :: TestDistImage (:)
REAL (KIND = 8), ALLOCATABLE :: TrainDistImage (:)

INTEGER (KIND = 2), ALLOCATABLE :: scx1 (:), icx1 (:)
INTEGER (KIND = 2), ALLOCATABLE :: scx2 (:), icx2 (:)
INTEGER (KIND = 2), ALLOCATABLE :: scx3 (:), icx3 (:)
```

```
 INTEGER (KIND = 2), ALLOCATABLE :: scx4 (:), icx4 (:)
 INTEGER (KIND = 2), ALLOCATABLE :: scx5 (:), icx5 (:)
 INTEGER (KIND = 2), ALLOCATABLE :: scx6 (:), icx6 (:)
 REAL (KIND = 8), ALLOCATABLE :: DistImage1B (:)
 REAL (KIND = 8), ALLOCATABLE :: DistImage2B (:)
 REAL (KIND = 8), ALLOCATABLE :: DistImage3B (:)
 REAL (KIND = 8), ALLOCATABLE :: DistImage4B (:)
 REAL (KIND = 8), ALLOCATABLE :: DistImage5B (:)
 REAL (KIND = 8), ALLOCATABLE :: DistImage6B (:)

INTEGER (KIND = 4) :: SkipNumber, SysOffset
INTEGER (KIND = 4) :: SysIndex
INTEGER (KIND = 4) :: TestSysIndex

 CHARACTER (LEN = S_LEN) :: InputFileNameImage
 CHARACTER (LEN = S_LEN) :: InputFileNameTrainingData
 CHARACTER (LEN = S_LEN) :: InputFileNameTrainingXY
 CHARACTER (LEN = S_LEN) :: InputFileNameValidationData
 CHARACTER (LEN = S_LEN) :: InputFileNamePlotProportion
 CHARACTER (LEN = S_LEN) :: InputFileNameValDataXY
 CHARACTER (LEN = S_LEN) :: OutputFileNameRoot
 CHARACTER (LEN = S_LEN) :: OutputFileNameValuesAll
 CHARACTER (LEN = S_LEN) :: OutputFileNameKSAll
 CHARACTER (LEN = S_LEN) :: OutputFileNameMeanVectors
 CHARACTER (LEN = S_LEN) :: OutputFileNameAAPoints
 CHARACTER (LEN = S_LEN) :: OutputFileNameXYDistance
 CHARACTER (LEN = S_LEN) :: OutputFileNameTrainDistA
 CHARACTER (LEN = S_LEN) :: OutputFileNameTrainDistB
 CHARACTER (LEN = S_LEN) :: OutputFileNameWAATrain
 CHARACTER (LEN = S_LEN) :: OutputFileNameAATrain
 CHARACTER (LEN = S_LEN) :: OutputFileNameTestDist
 CHARACTER (LEN = S_LEN) :: OutputFileNameAATest
 CHARACTER (LEN = S_LEN) :: OutputFileNameTestVectors
 CHARACTER (LEN = S_LEN) :: OutputFileNameAreaEst2
 CHARACTER (LEN = S_LEN) :: OutputFileNameTrValXYDist
 CHARACTER (LEN = S_LEN) :: OutputFileNameSpecDistTrain
 CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo1AA
 CHARACTER (LEN = S_LEN) :: OutputFileNameB1AAPoints
 CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo2AA
 CHARACTER (LEN = S_LEN) :: OutputFileNameB2AAPoints
 CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo3AA
 CHARACTER (LEN = S_LEN) :: OutputFileNameB3AAPoints
 CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo4AA
 CHARACTER (LEN = S_LEN) :: OutputFileNameB4AAPoints
 CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo5AA
 CHARACTER (LEN = S_LEN) :: OutputFileNameB5AAPoints
 CHARACTER (LEN = S_LEN) :: OutputFileNameBandCombo6AA
 CHARACTER (LEN = S_LEN) :: OutputFileNameB6AAPoints
 CHARACTER (LEN = S_LEN) :: OutputFileNameSpecDValTr
 CHARACTER (LEN = S_LEN) :: OutputFileNameSortAcc
 CHARACTER (LEN = S_LEN) :: OutputFileNameTrMinXYTr
 CHARACTER (LEN = S_LEN) :: OutputFileNameValMinXYTr
 CHARACTER (LEN = S_LEN) :: OutputFileNameValVectors
 CHARACTER (LEN = S_LEN) :: OutputFileNameTrainVectors

        open (10, file='sys_15.in', status = "old", action = "read")

        read (10, *) DrawSize
        PRINT *, "Draw Size: ", DrawSize

        read (10, *) InputFileNameImage
         PRINT *, "Input File Name Raw Image: ", InputFileNameImage

        read (10, *) k
        PRINT *, "Number of Pixels in Raw Image: ", k

        read (10, *) b
         PRINT *, "Number of Bands: ", b

        read (10, *) ImageNumNonzeroPixels
       PRINT *, "Number of Nonzero Pixels in Raw Image: ", ImageNumNonzeroPixels
```

```
        read (10, *) InputFileNameTrainingData
         PRINT *, "Input File Name Training Data: ", InputFileNameTrainingData

        read (10, *) InputFileNameTrainingXY
         PRINT *, "Input File Name Training XY: ", InputFileNameTrainingXY

        read (10, *) NumTrainingPoints
         PRINT *, "Number of Training Points: ", NumTrainingPoints

        read (10, *) InputFileNameValidationData
         PRINT *, "Input File Name Validation Data: ", InputFileNameValidationData

        read (10, *) InputFileNameValDataXY
        PRINT *, "Input File Name Validation XY: ", InputFileNameValDataXY

        read (10, *) InputFileNamePlotProportion
        PRINT *, "Input File Name Validation Data: ", InputFileNamePlotProportion

        read (10, *) NumValPoints
        PRINT *, "Number of Validation Points: ",  NumValPoints

        read (10, *) OutputFileNameRoot
        PRINT *, "Output File Name Root (no suffix, e.g. 1000x1000): ", &
               OutputFileNameRoot


OutputFileNameValuesAll = TRIM (OutputFileNameRoot) // "Values_All.txt"
OutputFileNameKSAll = TRIM (OutputFileNameRoot) // "KS_All.txt"
OutputFileNameMeanVectors = TRIM (OutputFileNameRoot) // "Mean_Vectors.txt"
 OutputFileNameAAPoints = TRIM (OutputFileNameRoot) // "AAPoints.txt"
OutputFileNameXYDistance = TRIM (OutputFileNameRoot) // "XYDistance.txt"
OutputFileNameTrainDistA = TRIM (OutputFileNameRoot) // "TrainDistA.txt"
OutputFileNameTrainDistB = TRIM (OutputFileNameRoot) // "TrainDistB.txt"
OutputFileNameWAATrain = TRIM (OutputFileNameRoot) // "WAATrain.txt"
OutputFileNameAATrain = TRIM (OutputFileNameRoot) // "AATrain.txt"
OutputFileNameTestDist = TRIM (OutputFileNameRoot) // "TestDist.txt"
OutputFileNameAATest = TRIM (OutputFileNameRoot) // "AATest.txt"
OutputFileNameTestVectors = TRIM (OutputFileNameRoot) // "Test_Vectors.txt"
OutputFileNameAreaEst2 = TRIM (OutputFileNameRoot) // "AreaEst2.txt"
OutputFileNameTrValXYDist = TRIM (OutputFileNameRoot) // "TrValXYDist.txt"
OutputFileNameSpecDistTrain = TRIM (OutputFileNameRoot) // "SpecDistTrain.txt"
OutputFileNameBandCombo1AA = TRIM (OutputFileNameRoot) // "BandCombo1AA.txt"
OutputFileNameB1AAPoints = TRIM (OutputFileNameRoot) // "B1AAPoints.txt"
OutputFileNameBandCombo2AA = TRIM (OutputFileNameRoot) // "BandCombo2AA.txt"
OutputFileNameB2AAPoints = TRIM (OutputFileNameRoot) // "B2AAPoints.txt"
OutputFileNameBandCombo3AA = TRIM (OutputFileNameRoot) // "BandCombo3AA.txt"
OutputFileNameB3AAPoints = TRIM (OutputFileNameRoot) // "B3AAPoints.txt"
OutputFileNameBandCombo4AA = TRIM (OutputFileNameRoot) // "BandCombo4AA.txt"
OutputFileNameB4AAPoints = TRIM (OutputFileNameRoot) // "B4AAPoints.txt"
OutputFileNameBandCombo5AA = TRIM (OutputFileNameRoot) // "BandCombo5AA.txt"
OutputFileNameB5AAPoints = TRIM (OutputFileNameRoot) // "B5AAPoints.txt"
OutputFileNameBandCombo6AA = TRIM (OutputFileNameRoot) // "BandCombo6AA.txt"
OutputFileNameB6AAPoints = TRIM (OutputFileNameRoot) // "B6AAPoints.txt"
OutputFileNameSpecDValTr = TRIM (OutputFileNameRoot) // "SpecDValTr.txt"
OutputFileNameSortAcc = TRIM (OutputFileNameRoot) // "SortAcc.txt"
OutputFileNameTrMinXYTr = TRIM (OutputFileNameRoot) // "TrMinXYTr.txt"
OutputFileNameValMinXYTr = TRIM (OutputFileNameRoot) // "ValMinXYTr.txt"
OutputFileNameValVectors = TRIM (OutputFileNameRoot) // "Val_Vectors.txt"
OutputFileNameTrainVectors = TRIM (OutputFileNameRoot) // "Train_Vectors.txt"

write (unit = *, FMT = *) "Input file names: "
write (unit = *, FMT = *) "    ", InputFileNameImage
write (unit = *, FMT = *) "    ", InputFileNameTrainingData
write (unit = *, FMT = *) "    ", InputFileNameValidationData
write (unit = *, FMT = *) "    ", InputFileNamePlotProportion
write (unit = *, FMT = *) "    ", InputFileNameTrainingXY
write (unit = *, FMT = *) "    ", InputFileNameValDataXY

write (unit = *, FMT = *) "Draw Size: "
write (unit = *, FMT = *) "    ", DrawSize
```

```
write (unit = *, FMT = *) "Number of Bands: "
write (unit = *, FMT = *) "    ", b, "(Raw Image)"

write (unit = *, FMT = *) "Number of Pixels in raw image: "
write (unit = *, FMT = *) "    ", k, " (Total)"

write (unit = *, FMT = *) "    ", ImageNumNonzeroPixels, " (Nonzero)"

write (unit = *, FMT = *) "Number of Validation Points: "
write (unit = *, FMT = *) "    ", NumValPoints

write (unit = *, FMT = *) "Output file names: "
write (unit = *, FMT = *) "    ", OutputFileNameValuesAll
write (unit = *, FMT = *) "    ", OutputFileNameKSAll
write (unit = *, FMT = *) "    ", OutputFileNameMeanVectors
write (unit = *, FMT = *) "    ", OutputFileNameAAPoints
write (unit = *, FMT = *) "    ", OutputFileNameXYDistance
write (unit = *, FMT = *) "    ", OutputFileNameTrainDistA
write (unit = *, FMT = *) "    ", OutputFileNameTrainDistB
write (unit = *, FMT = *) "    ", OutputFileNameWAATrain
write (unit = *, FMT = *) "    ", OutputFileNameAATrain
write (unit = *, FMT = *) "    ", OutputFileNameTestDist
write (unit = *, FMT = *) "    ", OutputFileNameAATest
write (unit = *, FMT = *) "    ", OutputFileNameTestVectors
write (unit = *, FMT = *) "    ", OutputFileNameAreaEst2
write (unit = *, FMT = *) "    ", OutputFileNameTrValXYDist
write (unit = *, FMT = *) "    ", OutputFileNameSpecDistTrain
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo1AA
write (unit = *, FMT = *) "    ", OutputFileNameB1AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo2AA
write (unit = *, FMT = *) "    ", OutputFileNameB2AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo3AA
write (unit = *, FMT = *) "    ", OutputFileNameB3AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo4AA
write (unit = *, FMT = *) "    ", OutputFileNameB4AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo5AA
write (unit = *, FMT = *) "    ", OutputFileNameB5AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameBandCombo6AA
write (unit = *, FMT = *) "    ", OutputFileNameB6AAPoints
write (unit = *, FMT = *) "    ", OutputFileNameSpecDValTr
write (unit = *, FMT = *) "    ", OutputFileNameSortAcc
write (unit = *, FMT = *) "    ", OutputFileNameTrMinXYTr
write (unit = *, FMT = *) "    ", OutputFileNameValMinXYTr
write (unit = *, FMT = *) "    ", OutputFileNameValVectors
write (unit = *, FMT = *) "    ", OutputFileNameTrainVectors

OneTenthNumDraws = 101
PRINT *, "OneTenthNumDraws = ", OneTenthNumDraws

TempTrDTr = 0.0
TempValDTr = 0.0
TestIndex = 0
TestSpecIndex = 0
Acres = 0.0
Xdiff = 0.0
Ydiff = 0.0
 TrainValXDiff = 0.0
TrainValYDiff = 0.0
TrValXdiff = 0.0
TrValYdiff = 0.0
TrValXYDistW = 0.0
TrXdiffTr = 0.0
TrYdiffTr = 0.0
TrainXDiff = 0.0
TrainYDiff = 0.0
TestXDiff = 0.0
TestYDiff = 0.0
XTrValDiff = 0.0
YTrValDiff = 0.0
BandsLessOne = 0
```

```
BandsLessTwo = 0
BandsLessThree = 0
BandsLessFour = 0
IPlusOne = 0
IPlusOneB = 0
IPlusOneC = 0
IPlusOneD = 0
JPlusOne = 0
JPlusOneB = 0
JPlusOneC = 0
LPlusOne = 0
LPlusOneB = 0
PPlusOne = 0
OnePlusJ = 0
NearestTrain = 0
NonforestStratVar = 0.0
ForestStratVar = 0.0
MeanPPForest = 0.0
MeanPPNonforest = 0.0

SingleRandomNumber = 0.0
TestRandomNumbers = 0.0
TestRandomIndices = 0
RandomIndices = 0
FirstRandomIndices = 0
SampleLocations = 0
SampleInfoClasses = 0
PercentAccuracy = 0.0
ErrorMatrix = 0
ForestAreaEstimate = 0.0
ForestMapMarginal = 0.0
StandardError = 0.0
scx = 0
icx = 0
ForestSampleSize = 0
NonforestSampleSize = 0
NumberWhole = 0

open (unit = 1, file = InputFileNameTrainingData, access = "sequential", &
       form = "formatted")

do i = 1, NumTrainingPoints

        read (unit = 1, fmt = "(I1)") InfoClasses(i)

end do

PRINT *, "Information classes read."


do i = 1, NumTrainingPoints

        read (unit = 1, fmt = "(I8)") Locations (i)

end do

close (unit = 1)

PRINT *, "Locations read."


open (unit = 22, file = InputFileNameTrainingXY, access = "sequential", &
       form = "formatted")

do i = 1, NumTrainingPoints

        read (unit = 22, fmt = "(F18.9)" ) XYTrain (i,1)

end do

do i = 1, NumTrainingPoints
```

```
        read (unit = 22, fmt = "(F18.9)" ) XYTrain (i,2)

end do

close (unit = 22)

PRINT *, "X and Y of training points read."


open (unit = 31, file = InputFileNameValDataXY, access = "sequential", &
       form = "formatted")

do i = 1, NumValPoints

        read (unit = 31, fmt = "(F18.9)" ) XYValData (i,1)

end do

do i = 1, NumValPoints

        read (unit = 31, fmt = "(F18.9)" ) XYValData (i,2)

end do

close (unit = 31)

PRINT *, "X and Y of validation points read."


open (unit = 11, file = InputFileNameImage, &
       form = "binary", status = "old", action = "read")

write (unit = *, fmt = *) "Reading input image into memory..."

ReadRawPixelsLoop: do i = 1, k

       ReadRawBandsLoop: do j = 1, b

               read (unit = 11) x (j,i) !, fmt = "(I)")

    end do ReadRawBandsLoop

end do ReadRawPixelsLoop


close (unit = 11)

write (unit = *, fmt = *) "Raw image written into memory..."


open (unit = 12, file = InputFileNameValidationData, access = "sequential", &
       form = "formatted")

PRINT *, "Reading validation data..."

ReadValidationValuesLoop: do i = 1, NumValPoints

        read (unit = 12, fmt = *) ValData (i,1)

end do ReadValidationValuesLoop

ReadValidationLocationsLoop: do i = 1, NumValPoints

        read (unit = 12, fmt = *) ValData (i,2)

end do ReadValidationLocationsLoop


close (unit = 12)
```

```
        PRINT *, "Reading plot proportions..."

        PlotProportionForest = 0.0

        open (unit = 30, file = InputFileNamePlotProportion, access = "sequential", &
               form = "formatted")

        ReadPlotProportionFLoop: do i = 1, NumValPoints

                read (unit = 30, fmt = "(F12.10)")  PlotProportionForest (i)

        end do ReadPlotProportionFLoop

        close (unit = 30)

!       Beginning of new steps for drawing systematic sample
        PRINT *, "Creating Samples, SampleInfoClasses, and SampleLocations..."

            SysOffset = 0
            SkipNumber = 0
            NumDraws = 0
            SysIndex = 0
            LeftOver = 0

            SkipNumber = (NumTrainingPoints / DrawSize)
            PRINT *, "SkipNumber = ", SkipNumber
            LeftOver = SkipNumber * DrawSize
            PRINT *, "LeftOver = ", LeftOver
            Remainder = NumTrainingPoints - LeftOver
            PRINT *, "Remainder = ", Remainder
            NumDraws = SkipNumber + Remainder
            PRINT *, "NumDraws = ", NumDraws

        DrawSystematicSamples: do i = 1, NumDraws


            SysIndex = i

            CreateSysSample: do j = 1, DrawSize

                SampleInfoClasses (j,i) = InfoClasses (SysIndex)
                SampleLocations (j,i) = Locations (SysIndex)
                SampleXYTrain (:,j,i) = XYTrain (SysIndex,:)
                RandomIndices (j,i) = SysIndex

                SysIndex = SysIndex + SkipNumber

            end do CreateSysSample

        end do DrawSystematicSamples


        PRINT *, "Creating Test samples..."

       call random_seed ()
        CALL random_number (TestRandomNumbers)


        FirstRandomIndices = int (TestRandomNumbers * NumDraws + 1, 4)

        call CompareTestnNumDraws(FirstRandomIndices, NumDraws)


        TestSysIndex = 0

        CreateTestSamples: do i = 1, NumDraws

        TestSysIndex = FirstRandomIndices(i)

        if (TestSysIndex .NE. i) then
```

```
        do j = 1, DrawSize

                TestInfoClasses (j,i) = InfoClasses (TestSysIndex)
                 TestLocations (j,i) = Locations (TestSysIndex)
                 TestXYTrain (:,j,i) = XYTrain (TestSysIndex,:)
                 TestRandomIndices (j,i) = TestSysIndex

               TestSysIndex = TestSysIndex + SkipNumber

         end do

  end if

  end do CreateTestSamples


  PRINT *, "Creating test validation data..."

  ALLOCATE (TestValData (DrawSize,2,NumDraws))

  TestValDataLoop: do i = 1, NumDraws

         do j = 1, DrawSize

        TestValData (j,1,i) = TestInfoClasses (j,i)
        TestValData (j,2,i) = j

         end do

  end do TestValDataLoop


  PRINT *, "Creating train validation data..."

  TrainValDataLoop: do i = 1, NumTrainingPoints

        TrainValData (i,1) = InfoClasses (i)
        TrainValData (i,2) = Locations (i)

  end do TrainValDataLoop


  TrainValDataBLoop: do i = 1, NumTrainingPoints

        TrainValDataB (i,1) = InfoClasses (i)
        TrainValDataB (i,2) = i

  end do TrainValDataBLoop


PRINT *, "Creating mean vectors"

  OuterMeanVectorsLoop: do i = 1, NumDraws

      InnerMeanVectorsLoop: do j = 1, DrawSize

          m (:, j, i) = x (:,SampleLocations (j,i))

      end do InnerMeanVectorsLoop

  end do OuterMeanVectorsLoop

  PRINT *, "Creating test vectors..."

  OuterTestVectorsLoop: do i = 1, NumDraws

      InnerTestVectorsLoop: do j = 1, DrawSize

          TestVector (:, j, i) = x (:, TestLocations (j,i))

      end do InnerTestVectorsLoop
```

```
        end do OuterTestVectorsLoop


        PRINT *, "Creating train vectors..."

        TrainingVectorsLoop: do i = 1, NumTrainingPoints

                TrainVector (:, i) = x (:, Locations (i))

        end do TrainingVectorsLoop

        open (unit = 50, file = OutputFileNameTrainVectors, access = "sequential", &
            form = "formatted", action = "write", status = "replace")

           WriteTrainVectors: do j = 1, NumTrainingPoints

            write (unit = 50,  fmt = "(I10, I10, I10, I10, I10, I10)") TrainVector (:,j)

            end do WriteTrainVectors

        close (unit = 50)

        PRINT *, "Creating validation vector..."

        ValidationVectorsLoop: do i = 1, NumValPoints

                ValVector (:, i) = x (:, ValData (i,2))

        end do ValidationVectorsLoop

        open (unit = 49,  file = OutputFileNameValVectors, access = "sequential", &
            form = "formatted", action = "write", status = "replace")

           WriteValVectors: do j = 1, NumValPoints

              write (unit = 49,  fmt = "(I10, I10, I10, I10, I10, I10)") ValVector (:,j)

            end do WriteValVectors

        close (unit = 49)

        Acres = (0.22239 * ImageNumNonzeroPixels)

        PRINT *, "Acres = ", Acres
        PRINT *, "Beginning band combinations, 1 band loop..."

! Exploring best subset of bands with all possible band combinations using just the
! validation data

       a = 1

        ErrorMatrix1b = 0
        PercentAccuracy1b = 0.0
        B1SpecAAPoints = 0
        B1SpecDistAA = 0.0
        m1 = 0.0
        x1 = 0

       OneBandLoop: do i = 1, b

                m1(1,:,:) = m(i,:,:)
                x1(1,:) = ValVector(i,:)


                        Classification1BLoop: do d = 1, NumDraws

                       ALLOCATE (scx1 (NumValPoints))
                       ALLOCATE (icx1 (NumValPoints))
                       ALLOCATE (DistImage1B (NumValPoints))
```

```
                    scx1 = 0
                    icx1 = 0
                     DistImage1B = 0.0

                      call nn (a, NumValPoints, DrawSize, x1, m1(:,:,d), scx1, &
                           DistImage1B)

                      call Recode (NumValPoints, scx1, SampleInfoClasses (:,d), icx1)

                      call ComputeAccuracyOnly (NumValPoints, icx1, NumValPoints, &
                           ValData, ErrorMatrix1b(i,d,:,:), PercentAccuracy1b(i,d), &
                           AAPoints1b(:,:,i,d))


                    do t = 1, NumValPoints

                            B1SpecAAPoints (t,1,i,d) = scx1 (t)
                            B1SpecDistAA (t,1,i,d) = DistImage1B (t)

                   end do

                   DEALLOCATE (scx1)
                   DEALLOCATE (icx1)
                   DEALLOCATE (DistImage1B)

                    end do Classification1BLoop

                    m1 = 0.0
                    x1 = 0

    end do OneBandLoop


    open (unit = 32,  file = OutputFileNameBandCombo1AA, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 33,  file = OutputFileNameB1AAPoints, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteB1Files: do i = 1, NumDraws

        WriteBandCombo1: do j = 1, 6

            write (unit = 32, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
           ErrorMatrix1b (j,i,1,1), ErrorMatrix1b (j,i,1,2), &
                  ErrorMatrix1b (j,i,2,1), ErrorMatrix1b (j,i,2,2), &
                  PercentAccuracy1b (j,i)

                  end do WriteBandCombo1

        WriteB1ComboAAPoints: do l = 1, 6

            do j = 1, NumValPoints

            write (unit = 33,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
            AAPoints1b (j,1,l,i), AAPoints1b (j,2,l,i), &
           B1SpecAAPoints (j,1,l,i), B1SpecDistAA (j,1,l,i)

            end do

        end do WriteB1ComboAAPoints

       end do WriteB1Files

close (unit = 32)
close (unit = 33)


    PRINT *, "Beginning band combinations, 2 band loop..."

    z = 0
```

```
c = 2

BandsLessOne = b - 1

ErrorMatrix2b = 0
PercentAccuracy2b = 0.0
B2SpecAAPoints = 0
B2SpecDistAA = 0.0
m2 = 0.0
x2 = 0

TwoBandLoop: do i = 1, BandsLessOne

                IPlusOne = i + 1

                Band2of2Loop: do j = IPlusOne, b

                       m2(1,:,:) = m(i,:,:)
                       m2(2,:,:) = m(j,:,:)
                      x2(1,:) = ValVector(i,:)
                      x2(2,:) = ValVector(j,:)

                z = z + 1

                Classification2BLoop: do d = 1, NumDraws


              ALLOCATE (scx2 (NumValPoints))
              ALLOCATE (icx2 (NumValPoints))
              ALLOCATE (DistImage2B (NumValPoints))

               scx2 = 0
               icx2 = 0
                DistImage2B = 0.0

                 call nn (c, NumValPoints, DrawSize, x2, m2(:,:,d), scx2, &
              DistImage2B)

                 call Recode (NumValPoints, scx2, SampleInfoClasses (:,d), icx2)

                 call ComputeAccuracyOnly (NumValPoints, icx2, NumValPoints, &
              ValData, ErrorMatrix2b(z,d,:,:), PercentAccuracy2b(z,d), &
       AAPoints2b(:,:,z,d))


               do t = 1, NumValPoints

                       B2SpecAAPoints (t,1,z,d) = scx2 (t)
                       B2SpecDistAA (t,1,z,d) = DistImage2B (t)

              end do

              DEALLOCATE (scx2)
              DEALLOCATE (icx2)
              DEALLOCATE (DistImage2B)

                end do Classification2BLoop

                m2 = 0.0
                x2 = 0

                end do Band2of2Loop

end do TwoBandLoop

open (unit = 34,  file = OutputFileNameBandCombo2AA, access = "sequential", &
    form = "formatted", action = "write", status = "replace")

open (unit = 35,  file = OutputFileNameB2AAPoints, access = "sequential", &
    form = "formatted", action = "write", status = "replace")
```

```
    WriteB2Files: do i = 1, NumDraws

        WriteBandCombo2: do j = 1, 15

            write (unit = 34, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
           ErrorMatrix2b (j,i,1,1), ErrorMatrix2b (j,i,1,2), &
                  ErrorMatrix2b (j,i,2,1), ErrorMatrix2b (j,i,2,2),&
                  PercentAccuracy2b (j,i)

                  end do WriteBandCombo2

        WriteB2ComboAAPoints: do l = 1, 15

            do j = 1, NumValPoints

            write (unit = 35,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
            AAPoints2b (j,1,l,i), AAPoints2b (j,2,l,i), &
       B2SpecAAPoints (j,1,l,i), B2SpecDistAA (j,1,l,i)

            end do

         end do WriteB2ComboAAPoints

        end do WriteB2Files

close (unit = 34)
close (unit = 35)


    PRINT *, "Beginning band combinations, 3 band loop..."

    y = 0
    e = 3

    BandsLessTwo = b - 2

    ErrorMatrix3b = 0
    PercentAccuracy3b = 0.0
    B3SpecAAPoints = 0
    B3SpecDistAA = 0.0
    m3 = 0.0
    x3 = 0

    ThreeBandLoop: do i = 1, BandsLessTwo

        IPlusOneB = i + 1

        Band2of3Loop: do j = IPlusOneB, BandsLessOne

            JPlusOne = j + 1

            Band3of3Loop: do l = JPlusOne, b

                m3(1,:,:) = m(i,:,:)
                m3(2,:,:) = m(j,:,:)
                m3(3,:,:) = m(l,:,:)
                 x3(1,:) = ValVector(i,:)
                 x3(2,:) = ValVector(j,:)
                 x3(3,:) = ValVector(l,:)

                     y = y + 1

                     Classification3BLoop: do d = 1, NumDraws


                   ALLOCATE (scx3 (NumValPoints))
                   ALLOCATE (icx3 (NumValPoints))
                   ALLOCATE (DistImage3B (NumValPoints))

                    scx3 = 0
                    icx3 = 0
```

```
                DistImage3B = 0.0

                    call nn (e, NumValPoints, DrawSize, x3, m3(:,:,d), scx3, &
                         DistImage3B)

                    call Recode (NumValPoints, scx3, SampleInfoClasses (:,d), icx3)

                    call ComputeAccuracyOnly (NumValPoints, icx3, NumValPoints, &
                          ValData, ErrorMatrix3b(y,d,:,:), PercentAccuracy3b(y,d), &
                          AAPoints3b(:,:,y,d))


                  do t = 1, NumValPoints

                  B3SpecAAPoints (t,1,y,d) = scx3 (t)
                          B3SpecDistAA (t,1,y,d) = DistImage3B (t)

                 end do

                 DEALLOCATE (scx3)
                 DEALLOCATE (icx3)
                 DEALLOCATE (DistImage3B)

                   end do Classification3BLoop

                   m3 = 0.0
                   x3 = 0

            end do Band3of3Loop

        end do Band2of3Loop

    end do ThreeBandLoop


    open (unit = 36,  file = OutputFileNameBandCombo3AA, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 37,  file = OutputFileNameB3AAPoints, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteB3Files: do i = 1, NumDraws

        WriteBandCombo3: do j = 1, 20

            write (unit = 36, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
                   ErrorMatrix3b (j,i,1,1), ErrorMatrix3b (j,i,1,2), &
                   ErrorMatrix3b (j,i,2,1), ErrorMatrix3b (j,i,2,2),&
                   PercentAccuracy3b (j,i)

                  end do WriteBandCombo3

        WriteB3ComboAAPoints: do l = 1, 20

            do j = 1, NumValPoints

            write (unit = 37,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
            AAPoints3b (j,1,l,i), AAPoints3b (j,2,l,i), &
           B3SpecAAPoints (j,1,l,i), B3SpecDistAA (j,1,l,i)

            end do

        end do WriteB3ComboAAPoints

       end do WriteB3Files

close (unit = 36)
close (unit = 37)


    PRINT *, "Beginning band combinations, 4 band loop..."
```

```
w = 0
f = 4

BandsLessThree = b - 3

ErrorMatrix4b = 0
PercentAccuracy4b = 0.0
B4SpecAAPoints = 0
B4SpecDistAA = 0.0
m4 = 0.0
x4 = 0

FourBandLoop: do i = 1, BandsLessThree

    IPlusOneC = i + 1

    Band2of4Loop: do j = IPlusOneC, BandsLessTwo

        JPlusOneB = j + 1

        Band3of4Loop: do l = JPlusOneB, BandsLessOne

            LPlusOne = l + 1

            Band4of4Loop: do p = LPlusOne, b

           m4(1,:,:) = m(i,:,:)
           m4(2,:,:) = m(j,:,:)
           m4(3,:,:) = m(l,:,:)
           m4(4,:,:) = m(p,:,:)
            x4(1,:) = ValVector(i,:)
            x4(2,:) = ValVector(j,:)
            x4(3,:) = ValVector(l,:)
            x4(4,:) = ValVector(p,:)

                w = w + 1

                Classification4BLoop: do d = 1, NumDraws


              ALLOCATE (scx4 (NumValPoints))
              ALLOCATE (icx4 (NumValPoints))
              ALLOCATE (DistImage4B (NumValPoints))

               scx4 = 0
              icx4 = 0
                DistImage4B = 0.0

                 call nn (f, NumValPoints, DrawSize, x4, m4(:,:,d), scx4, &
                       DistImage4B)

                 call Recode (NumValPoints, scx4, SampleInfoClasses (:,d), icx4)

                 call ComputeAccuracyOnly (NumValPoints, icx4, NumValPoints, &
                       ValData, ErrorMatrix4b(w,d,:,:), PercentAccuracy4b(w,d), &
                       AAPoints4b(:,:,w,d))


               do t = 1, NumValPoints

                B4SpecAAPoints (t,1,w,d) = scx4 (t)
                       B4SpecDistAA (t,1,w,d) = DistImage4B (t)

              end do

              DEALLOCATE (scx4)
              DEALLOCATE (icx4)
              DEALLOCATE (DistImage4B)

               end do Classification4BLoop
```

```
                    m4 = 0.0
                    x4 = 0

                end do Band4of4Loop

            end do Band3of4Loop

        end do Band2of4Loop

    end do FourBandLoop


    open (unit = 38,  file = OutputFileNameBandCombo4AA, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 39,  file = OutputFileNameB4AAPoints, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteB4Files: do i = 1, NumDraws

        WriteBandCombo4: do j = 1, 15

            write (unit = 38, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
                  ErrorMatrix4b (j,i,1,1), ErrorMatrix4b (j,i,1,2), &
                  ErrorMatrix4b (j,i,2,1), ErrorMatrix4b (j,i,2,2),&
           PercentAccuracy4b (j,i)

                  end do WriteBandCombo4

        WriteB4ComboAAPoints: do l = 1, 15

            do j = 1, NumValPoints

            write (unit = 39,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
            AAPoints4b (j,1,l,i), AAPoints4b (j,2,l,i), &
           B4SpecAAPoints (j,1,l,i), B4SpecDistAA (j,1,l,i)

            end do

        end do WriteB4ComboAAPoints

       end do WriteB4Files

close (unit = 38)
close (unit = 39)


    PRINT *, "Beginning band combinations, 5 band loop..."

    v = 0
    g = 5

    BandsLessFour = b - 4

    ErrorMatrix5b = 0
    PercentAccuracy5b = 0.0
    B5SpecAAPoints = 0
    B5SpecDistAA = 0.0
    m5 = 0.0
    x5 = 0

   FiveBandLoop: do i = 1, BandsLessFour

        IPlusOneD = i + 1

        Band2of5Loop: do j = IPlusOneD, BandsLessThree

            JPlusOneC = j + 1

            Band3of5Loop: do l = JPlusOneC, BandsLessTwo
```

```
            LPlusOneB = l + 1

            Band4of5Loop: do p = LPlusOneB, BandsLessOne

                PPlusOne = p + 1

                Band5of5Loop: do q = PPlusOne, b

                   m5(1,:,:) = m(i,:,:)
                   m5(2,:,:) = m(j,:,:)
                   m5(3,:,:) = m(l,:,:)
                   m5(4,:,:) = m(p,:,:)
                   m5(5,:,:) = m(q,:,:)
                    x5(1,:) = ValVector(i,:)
                    x5(2,:) = ValVector(j,:)
                    x5(3,:) = ValVector(l,:)
                    x5(4,:) = ValVector(p,:)
                    x5(5,:) = ValVector(q,:)

                v = v + 1

                Classification5BLoop: do d = 1, NumDraws


              ALLOCATE (scx5 (NumValPoints))
              ALLOCATE (icx5 (NumValPoints))
              ALLOCATE (DistImage5B (NumValPoints))

               scx5 = 0
               icx5 = 0
             DistImage5B = 0.0

                 call nn (g, NumValPoints, DrawSize, x5, m5(:,:,d), scx5, &
              DistImage5B)

                 call Recode (NumValPoints, scx5, SampleInfoClasses (:,d), icx5)

                 call ComputeAccuracyOnly (NumValPoints, icx5, NumValPoints, &
                       ValData, ErrorMatrix5b(v,d,:,:), PercentAccuracy5b(v,d), &
                       AAPoints5b(:,:,v,d))


               do t = 1, NumValPoints

                       B5SpecAAPoints (t,1,v,d) = scx5 (t)
                       B5SpecDistAA (t,1,v,d) = DistImage5B (t)

              end do

              DEALLOCATE (scx5)
              DEALLOCATE (icx5)
              DEALLOCATE (DistImage5B)

               end do Classification5BLoop

               m5 = 0.0
               x5 = 0

               end do Band5of5Loop

            end do Band4of5Loop

        end do Band3of5Loop

    end do Band2of5Loop

end do FiveBandLoop


open (unit = 40,  file = OutputFileNameBandCombo5AA, access = "sequential", &
```

```
          form = "formatted", action = "write", status = "replace")

      open (unit = 41,  file = OutputFileNameB5AAPoints, access = "sequential", &
          form = "formatted", action = "write", status = "replace")

     WriteB5Files: do i = 1, NumDraws

          WriteBandCombo5: do j = 1, 6

              write (unit = 40, fmt = "(I10,I10,I10,I10,I10,I10,F12.6)") i, j, &
                    ErrorMatrix5b (j,i,1,1), ErrorMatrix5b (j,i,1,2), &
                     ErrorMatrix5b (j,i,2,1), ErrorMatrix5b (j,i,2,2),&
              PercentAccuracy5b (j,i)

                    end do WriteBandCombo5

          WriteB5ComboAAPoints: do l = 1, 6

             do j = 1, NumValPoints

              write (unit = 41,  fmt = "(I10,I10,I12,I10,I10,F12.3)") i, l, &
              AAPoints5b (j,1,l,i), AAPoints5b (j,2,l,i), &
             B5SpecAAPoints (j,1,l,i), B5SpecDistAA (j,1,l,i)

              end do

          end do WriteB5ComboAAPoints

         end do WriteB5Files

close (unit = 40)
close (unit = 41)


      PRINT *, "Beginning band combinations, 6 band loop..."

     ErrorMatrix6b = 0
     PercentAccuracy6b = 0
     B6SpecAAPoints = 0
     B6SpecDistAA = 0.0

     Classification6BLoop: do d = 1, NumDraws

     ALLOCATE (scx6 (NumValPoints))
     ALLOCATE (icx6 (NumValPoints))
     ALLOCATE (DistImage6B (NumValPoints))

      scx6 = 0
      icx6 = 0
      DistImage6B = 0.0

     call nn (b, NumValPoints, DrawSize, ValVector, m(:,:,d), scx6, DistImage6B)

     call Recode (NumValPoints, scx6, SampleInfoClasses (:,d), icx6)

     call ComputeAccuracyOnly (NumValPoints, icx6, NumValPoints, ValData, &
           ErrorMatrix6b(d,:,:), PercentAccuracy6b(d), AAPoints6b(:,:,d))

             do t = 1, NumValPoints

                     B6SpecAAPoints (t,1,d) = scx6 (t)
                     B6SpecDistAA (t,1,d) = DistImage6B (t)

             end do

     DEALLOCATE (scx6)
     DEALLOCATE (icx6)
     DEALLOCATE (DistImage6B)

     end do Classification6BLoop
```

```
    PRINT *, "Sorting accuracies from six band loop."


    Sort6bClassificationLoop: do i = 1, NumDraws

    SortAccuracy(1,i) = i
    SortAccuracy(2,i) = ErrorMatrix6b (i,1,1) + ErrorMatrix6b (i,2,2)

    end do Sort6bClassificationLoop


    call QuickSort(SortAccuracy, 1, NumDraws)


     open (unit = 46,  file = OutputFileNameSortAcc, access = "sequential", &
         form = "formatted", action = "write", status = "replace")

     WriteSortFile: do i = 1, NumDraws

     write (unit = 46, fmt = "(I10,I10)") SortAccuracy (1,i), SortAccuracy (2,i)

        end do WriteSortFile

close (unit = 46)

     PRINT *, "Beginning draw specific processing loop..."

     ForestMapMarginal = 0.0
     StandardError = 0.0
     PrecisionFAE = 0.0
     PercentAccuracy = 0.0
     TotalClassForest = 0
     TotalClassNonforest = 0
     ErrorMatrix = 0
     ForestAreaEstimate = 0.0
     PercentAccuracyTr = 0.0
     ErrorMatrixTr = 0
     ForestAreaEstimateTr = 0.0
     StandardErrorTr = 0.0
     SpecDistAA = 0.0
     SpecAAPoints = 0
     OneTenthCount = 0
     DrawSkipNumber = 0.0
     IEdit = 0
     NumDrawsReal = 0.0

     NumDrawsReal = NumDraws

     if (NumDraws .GT. 101) then

     DrawSkipNumber = NumDrawsReal/100
     PRINT *, "DrawSkipNumber = ", DrawSkipNumber

     ComputeIEdit: do i = 1, 101

     if (i .EQ. 1) then
     IEdit(i) = i
     PRINT *, IEdit(i)

     else
     IEdit(i) = (i-1) * DrawSkipNumber
     PRINT *, IEdit(i)
     end if

     end do ComputeIEdit

     else
     DrawSkipNumber = 1.0

     ComputeIEdit2: do i = 1, NumDraws
```

```
if (i .EQ. 1) then
IEdit(i) = i
PRINT *, IEdit(i)

else
IEdit(i) = (i-1) * DrawSkipNumber
PRINT *, IEdit(i)
end if

end do ComputeIEdit2
end if

CalculateWhole: if (NumDraws .GT. 101) then

        NumberWhole = 101
        else
        NumberWhole = NumDraws

                end if CalculateWhole


DrawSpecificProcessingLoop: do i = 1, NumberWhole

        PRINT *, "Sort number = ", IEdit(i)

        OneTenthCount = OneTenthCount + 1

       ALLOCATE (scx (k))
       scx = 0
       ALLOCATE (icx (k))
       icx = 0
        ALLOCATE (DistanceImage (k))
       DistanceImage = 0.0

    PRINT *, "Nearest neighbor computation, draw ", IEdit(i)

    SampleNumber = SortAccuracy(1,IEdit(i))

    WImageDrawNum (OneTenthCount) = SampleNumber

    call nn (b, k, DrawSize, x, m(:,:,SampleNumber), scx, DistanceImage)

    PRINT *, "Recode, draw ", IEdit(i)

    call Recode (k, scx, SampleInfoClasses (:,SampleNumber), icx)

    PRINT *, "Computing accuracy plus area estimate, draw ", IEdit(i)

    CALL AccPlusAreaEst (k, icx, NumValPoints, ValData, Acres,&
         ErrorMatrix(OneTenthCount,:,:), ForestAreaEstimate(OneTenthCount), &
         ForestMapMarginal(OneTenthCount), PercentAccuracy(OneTenthCount), &
         StandardError(OneTenthCount), PrecisionFAE(OneTenthCount), &
         AAPoints (:,:,OneTenthCount))

TotalClassForest (OneTenthCount) = ErrorMatrix (OneTenthCount,1,2) + &
       ErrorMatrix (OneTenthCount,2,2)

TotalClassNonforest (OneTenthCount) = ErrorMatrix (OneTenthCount,1,1) + &
        ErrorMatrix (OneTenthCount,2,1)

        do j = 1, NumValPoints

            SpecAAPoints (j,1,OneTenthCount) = scx (ValData (j,2))
                 SpecDistAA (j,1,OneTenthCount) = DistanceImage (ValData (j,2))
                 NumTrPointW (j,1,OneTenthCount) = &
                      RandomIndices (scx (ValData (j,2)),SampleNumber)
                 TrValXdiff = XYValData (j,1) - &
                      SampleXYTrain (1,scx(ValData(j,2)),SampleNumber)
                 TrValYdiff = XYValData (j,2) - &
                      SampleXYTrain (2,scx(ValData(j,2)),SampleNumber)
```

```
                TrValXYDistW = SQRT ((TrValXdiff*TrValXdiff) + &
       (TrValYdiff*TrValYdiff))
                SpatialDistW (j,1,OneTenthCount) = TrValXYDistW
       end do

    PRINT *, "Computing McRoberts's forest area estimate, draw ", IEdit(i)

CALL ComputeMcRobertsAreaEstimate (k, icx, NumValPoints, ValData, &
ForestMapMarginal(OneTenthCount), PlotProportionForest, Acres, &
TotalClassForest(OneTenthCount), TotalClassNonforest(OneTenthCount), &
MeanPPForest(OneTenthCount), MeanPPNonforest(OneTenthCount), &
WeightedProportion (OneTenthCount), ForestArea (OneTenthCount), &
ForestStratVar(OneTenthCount), NonforestStratVar(OneTenthCount), &
VarianceSum (OneTenthCount), VarianceForestArea (OneTenthCount), &
FAEPrecision (OneTenthCount))

    PRINT *, "Computing accuracy plus area estimate train, draw ", IEdit(i)

    CALL AccPlusAreaEstP2 (k, icx, NumTrainingPoints, TrainValData, &
       ForestMapMarginal(OneTenthCount), ErrorMatrixTr(OneTenthCount,:,:), &
        ForestAreaEstimateTr(OneTenthCount), PercentAccuracyTr(OneTenthCount), &
        StandardErrorTr(OneTenthCount), AAPointsTr (:,:,OneTenthCount))

     DEALLOCATE (scx)
    DEALLOCATE (icx)
    DEALLOCATE (DistanceImage)


end do DrawSpecificProcessingLoop


open (unit = 4, file = OutputFileNameKSAll, access = "sequential", &
       form = "formatted", action = "write", status = "replace")

open (unit = 16,  file = OutputFileNameAAPoints, access = "sequential", &
    form = "formatted", action = "write", status = "replace")

open (unit = 19,  file = OutputFileNameWAATrain, access = "sequential", &
    form = "formatted", action = "write", status = "replace")

open (unit = 26,  file = OutputFileNameAreaEst2, access = "sequential", &
    form = "formatted", action = "write", status = "replace")

WriteFilesB: do i = 1, OneTenthNumDraws

    WriteAAPoints: do j = 1, NumValPoints

        write (unit = 16,  fmt = "(I10,I12,I10,I10,F12.3,I10,F15.5)") i, &
              AAPoints(j,1,i), AAPoints(j,2,i), SpecAAPoints (j,1,i), &
              SpecDistAA (j,1,i), NumTrPointW(j,1,i), SpatialDistW(j,1,i)

    end do WriteAAPoints

    write (unit = 4, fmt = &
       "(I10,I10,I10,I10,I10,F12.8,F12.8,F12.6,ES20.9,ES20.9,I10,I10)") &
        WImageDrawNum(i), ErrorMatrix(i,1,1), ErrorMatrix(i,1,2), &
              ErrorMatrix(i,2,1), ErrorMatrix(i,2,2), ForestAreaEstimate(i), &
              ForestMapMarginal(i), PercentAccuracy(i), StandardError(i), &
              PrecisionFAE(i), TotalClassForest (i), TotalClassNonforest(i)

    write (unit = 19, fmt = "(I10,I10,I10,I10,F12.8,F12.8,F12.6,ES20.9)") &
        ErrorMatrixTr(i,1,1), ErrorMatrixTr(i,1,2), ErrorMatrixTr(i,2,1), &
        ErrorMatrixTr(i,2,2), ForestAreaEstimateTr(i), ForestMapMarginal(i), &
        PercentAccuracyTr(i), StandardErrorTr(i)

    write (unit = 26, fmt = &
       "(F12.8,F12.8,F12.8,F15.5,F12.8,F12.8,ES20.9,F20.5,ES20.9,I10,I10)")&
        MeanPPForest(i), MeanPPNonforest(i), WeightedProportion(i), &
              ForestArea(i), ForestStratVar(i), NonforestStratVar(i), &
              VarianceSum(i), VarianceForestArea(i), FAEPrecision(i), &
              TotalClassForest(i), TotalClassNonforest(i)
```

```
        end do WriteFilesB

close (unit = 4)
close (unit = 16)
close (unit = 19)
close (unit = 26)

    ALLOCATE (TestAAPoints (DrawSize, 2, NumDraws))

    TestErrorMatrix = 0
    TestForestAreaEstimate = 0.0
    TestPercentAccuracy = 0.0
    TestForestMapMarginal = 0.0
    TestStandardError = 0.0
    SpecDistTest = 0.0
    TestPrecisionFAE = 0.0

    TestDrawProcessingLoop: do i = 1, NumDraws

           ALLOCATE (Testscx (DrawSize))
            Testscx = 0
           ALLOCATE (Testicx (DrawSize))
            Testicx = 0
             ALLOCATE (TestDistImage (DrawSize))
            TestDistImage = 0.0

        PRINT *, "Nearest neighbor test computation, draw ", i

        call nn (b, DrawSize, DrawSize, TestVector(:,:,i), m(:,:,i), Testscx, &
           TestDistImage)

        PRINT *, "Recode test, draw ", i

        call Recode (DrawSize, Testscx, SampleInfoClasses (:, i), Testicx)

        PRINT *, "Computing accuracy plus area estimate test, draw ", i

        CALL AccPlusAreaEst (DrawSize, Testicx, DrawSize, TestValData (:,:,i), &
            Acres, TestErrorMatrix(i,:,:), TestForestAreaEstimate(i), &
            TestForestMapMarginal(i), TestPercentAccuracy(i), TestStandardError(i), &
            TestPrecisionFAE(i), TestAAPoints (:,:,i))

           do j = 1, DrawSize

           TestXDiff = 0.0
           TestYDiff = 0.0

                    NumTrPtTest (j,1,i) = TestRandomIndices (j,i)
                    SpecClassTest (j,1,i) = Testscx (j)
                    SpecDistTest (j,1,i) = TestDistImage (j)
                    NumTrPtSpecTest (j,1,i) = TestRandomIndices (Testscx(j), i)
                    TestXDiff = TestXYTrain (1,j,i) - SampleXYTrain (1,Testscx(j),i)
                    TestYDiff = TestXYTrain (2,j,i) - SampleXYTrain (2,Testscx(j),i)
                    SpatialDTest (j,1,i) = SQRT((TestXDiff*TestXDiff) + &
                           (TestYDiff*TestYDiff))

           end do

         DEALLOCATE (Testscx)
        DEALLOCATE (Testicx)
        DEALLOCATE (TestDistImage)

    end do TestDrawProcessingLoop

DEALLOCATE (TestValData)

    PRINT *, "Writing test files..."

    open (unit = 24,  file = OutputFileNameTestVectors, access = "sequential", &
     form = "formatted", action = "write", status = "replace")
```

```
    open (unit = 23,  file = OutputFileNameAATest, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 25,  file = OutputFileNameTestDist, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteTestFiles: do i = 1, NumDraws

        WriteTestVectors: do j = 1, DrawSize

            write (unit = 24,  fmt = "(I10, I10, I10, I10, I10, I10)") &
                   TestVector (:,j,i)

                        end do WriteTestVectors
        write (unit = 24, fmt = "(I10, I10, I10, I10, I10, I10)") &
                  -1, -1, -1, -1, -1,-1

        write (unit = 23, fmt ="(I10,I10,I10,I10,F12.8,F12.8,F12.6,ES20.9,ES20.9)")&
    TestErrorMatrix(i,1,1), TestErrorMatrix(i,1,2), TestErrorMatrix(i,2,1), &
         TestErrorMatrix(i,2,2), TestForestAreaEstimate(i), &
           TestForestMapMarginal(i), TestPercentAccuracy(i), TestStandardError(i), &
           TestPrecisionFAE(i)

        WriteSpecTestDist: do j = 1, DrawSize

        write (unit = 25, fmt = "(I10,I10,I10,I10,I10,F11.3,I10,F15.5)") i, &
           TestAAPoints (j,1,i), TestAAPoints (j,2,i), NumTrPtTest(j,1,i), &
           SpecClassTest (j,1,i), SpecDistTest (j,1,i), NumTrPtSpecTest(j,1,i), &
           SpatialDTest(j,1,i)

                         end do WriteSpecTestDist

   end do WriteTestFiles

close (unit = 24)
close (unit = 23)
close (unit = 25)

DEALLOCATE (TestAAPoints)

    PRINT *, "Train processing loop..."

    TrainPercentAccuracy = 0.0
    TrainForestMapMarginal = 0.0
    TrainStandardError = 0.0
    TrainForestAreaEstimate = 0.0
    SpatialDistTr = 0.0
    SpecDistTrain = 0.0
    TrainPrecisionFAE = 0.0

    TrainDrawProcessingLoop: do i = 1, NumDraws

           ALLOCATE (Trainscx (NumTrainingPoints))
            Trainscx = 0
           ALLOCATE (Trainicx (NumTrainingPoints))
            Trainicx = 0
            ALLOCATE (TrainDistImage (NumTrainingPoints))
           TrainDistImage = 0.0

        PRINT *, "Nearest neighbor train computation, draw ", i

        call nn (b, NumTrainingPoints, DrawSize, TrainVector, m(:,:,i), Trainscx, &
           TrainDistImage)

        PRINT *, "Recode train, draw ", i

        call Recode (NumTrainingPoints, Trainscx, SampleInfoClasses(:,i), Trainicx)

        PRINT *, "Computing accuracy plus area estimate train, draw ", i
```

```
        CALL AccPlusAreaEst (NumTrainingPoints, Trainicx, NumTrainingPoints, &
             TrainValDataB, Acres, TrainErrorMatrix(i,:,:), &
                  TrainForestAreaEstimate(i), TrainForestMapMarginal(i), &
                  TrainPercentAccuracy(i), TrainStandardError(i), &
                  TrainPrecisionFAE(i), TrainAAPoints (:,:,i))

            do j = 1, NumTrainingPoints

            TrainXDiff = 0.0
            TrainYDiff = 0.0

                     SpecClassTrain (j,1,i) = Trainscx (j)
                     SpecDistTrain (j,1,i) = TrainDistImage (j)
                     NumTrPtTrain (j,1,i) = RandomIndices (Trainscx(j),i)
                     TrainXDiff = XYTrain (j,1) - SampleXYTrain (1,Trainscx(j),i)
                     TrainYDiff = XYTrain (j,2) - SampleXYTrain (2,Trainscx(j),i)
                     SpatialDistTr (j,1,i) = SQRT((TrainXDiff*TrainXDiff) + &
           (TrainYDiff*TrainYDiff))
           end do

         DEALLOCATE (Trainscx)
         DEALLOCATE (Trainicx)
         DEALLOCATE (TrainDistImage)

    end do TrainDrawProcessingLoop


    PRINT *, "Writing mean vectors..."

    open (unit = 15,  file = OutputFileNameMeanVectors, access = "sequential", &
     form = "formatted", action = "write", status = "replace")

    WriteMeanFiles: do i = 1, NumDraws

            MeanVectors: do j = 1, DrawSize

             write (unit = 15,  fmt = "(F10.1, F10.1, F10.1, F10.1, F10.1, F10.1)") &
                    m (:,j,i)

        end do MeanVectors

        write (unit = 15, fmt = "(I10, I10, I10, I10, I10, I10)") &
            -1, -1, -1, -1, -1,-1

       end do WriteMeanFiles

close (unit = 15)


    PRINT *, "Writing train files..."

    open (unit = 44,  file = OutputFileNameAATrain, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 18,  file = OutputFileNameTrainDistA, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 51,  file = OutputFileNameTrainDistB, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteTrainFiles: do i = 1, NumDraws

        write (unit = 44, fmt ="(I10,I10,I10,I10,F12.8,F12.8,F12.6,ES20.9,ES20.9)")&
           TrainErrorMatrix(i,1,1), TrainErrorMatrix(i,1,2), &
           TrainErrorMatrix(i,2,1), TrainErrorMatrix(i,2,2), &
           TrainForestAreaEstimate(i), TrainForestMapMarginal(i), &
           TrainPercentAccuracy(i), TrainStandardError(i), TrainPrecisionFAE(i)

   end do WriteTrainFiles

   HalfDraws = NumDraws/2
```

```
   PRINT *, "HalfDraws = ", HalfDraws

        WriteSpecTrDA: do i = 1, HalfDraws

        WriteSpecTrainDistA: do j = 1, NumTrainingPoints

        write (unit = 18, fmt = "(I10,I10,I10,I10,F11.3,I10,F15.5)") i, &
            TrainAAPoints (j,1,i), TrainAAPoints (j,2,i), SpecClassTrain (j,1,i), &
            SpecDistTrain (j,1,i), NumTrPtTrain (j,1,i), SpatialDistTr(j,1,i)

                end do WriteSpecTrainDistA

        end do WriteSpecTrDA


        WriteSpecTrDB: do i = HalfDraws+1, NumDraws

        WriteSpecTrainDistB: do j = 1, NumTrainingPoints

        write (unit = 51, fmt = "(I10,I10,I10,I10,F11.3,I10,F15.5)") i, &
           TrainAAPoints (j,1,i), TrainAAPoints (j,2,i), SpecClassTrain (j,1,i), &
           SpecDistTrain (j,1,i), NumTrPtTrain (j,1,i), SpatialDistTr(j,1,i)

                end do WriteSpecTrainDistB

        end do WriteSpecTrDB

close (unit = 44)
close (unit = 18)
close (unit = 51)


    PRINT *, "Calculating sample sizes by information class..."

    ForestSampleSize = 0
    NonforestSampleSize = 0

   ComputeSampleSizes: do i = 1, NumDraws

          do j = 1, DrawSize

                if (SampleInfoClasses (j,i) == 1) then

                        NonforestSampleSize (i) = NonforestSampleSize (i) + 1

                end if

                if (SampleInfoClasses (j,i) == 2) then

                        ForestSampleSize (i) = ForestSampleSize (i) + 1

                end if

           end do

end do ComputeSampleSizes

    PRINT *, "Writing values all file..."

    open (unit = 3, file = OutputFileNameValuesAll, access = "sequential", &
          form = "formatted", action = "write", status = "replace")

    WriteSFiles: do i = 1, NumDraws

        WriteSamples: do j = 1, DrawSize

            write (unit = 3, fmt = &
                "(I10,I10,I5,F18.9,F18.9,I10,I10,I5,F18.9,F18.9)")  &
              RandomIndices (j,i), SampleLocations (j,i), &
                SampleInfoClasses (j,i), SampleXYTrain (1,j,i), &
                SampleXYTrain (2,j,i), TestRandomIndices (j,i), &
```

```
                TestLocations (j,i), TestInfoClasses (j,i), TestXYTrain (1,j,i), &
                  TestXYTrain (2,j,i)

        end do WriteSamples

        write (unit = 3, fmt = "(I10,I10,I10,I10,I10,I10,I10,I10,I10,I10)") &
            -1,-1,-1,-1,-1,-1,-1,-1,-1,-1

    end do WriteSFiles

close (unit = 3)


    PRINT *, "Writing 6 band combination files..."

    open (unit = 42,  file = OutputFileNameBandCombo6AA, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    open (unit = 43,  file = OutputFileNameB6AAPoints, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteB6Files: do i = 1, NumDraws

            write (unit = 42, fmt = "(I10,I10,I10,I10,I10,F12.6,I10,I10)") i, &
                  ErrorMatrix6b(i,1,1), ErrorMatrix6b(i,1,2), ErrorMatrix6b (i,2,1), &
                  ErrorMatrix6b (i,2,2), PercentAccuracy6b (i), &
                  NonforestSampleSize(i), ForestSampleSize(i)


        WriteB6ComboAAPoints: do j = 1, NumValPoints

            write (unit = 43,  fmt = "(I10,I12,I10,I10,F12.3)") i, &
            AAPoints6b (j,1,i), AAPoints6b (j,2,i), &
           B6SpecAAPoints (j,1,i), B6SpecDistAA (j,1,i)

        end do WriteB6ComboAAPoints

       end do WriteB6Files

close (unit = 42)
close (unit = 43)


    PRINT *, "Calculating spatial distance between training points..."

    TrminusTr = 0.0

    FindClosestSpatialTr2Tr: do i = 1, NumDraws

    do j = 1, DrawSize

   MinTrXYDist = 1000000.0
    TrNearestTrain = 0

           do l = 1, DrawSize

            if (l .NE. j) then


                  if (RandomIndices(l,i) .NE. RandomIndices(j,i)) then

                       TrXdiffTr = XYTrain(RandomIndices (j,i),1) - &
                                 XYTrain(RandomIndices (l,i),1)
                       TrYdiffTr = XYTrain(RandomIndices (j,i),2) - &
                                 XYTrain(RandomIndices (l,i),2)
                       TrDistTr = SQRT((TrXdiffTr*TrXdiffTr)+(TrYdiffTr*TrYdiffTr))
                       if (TrDistTr .LT. MinTrXYDist) then

                              MinTrXYDist = TrDistTr
                            TrNearestTrain = RandomIndices(l,i)
```

```
                        end if

                end if

        end if

        end do
                TrMinXY2TrainA (j,1,i) = RandomIndices(j,i)
                TrMinXY2Train (j,1,i) = MinTrXYDist
                TrMinXY2TrainA (j,2,i) = TrNearestTrain

                        TrminusTr (:,1) = TrainVector(:,RandomIndices(j,i)) - &
                                TrainVector(:,TrNearestTrain)

                OnebyOneMatrix = &
                                SQRT (MATMUL (TRANSPOSE (TrminusTr), &
                                   TrminusTr))

                TempTrDTr = OnebyOneMatrix (1,1)
                TrMinXY2Train (j,2,i) = TempTrDTr

        end do

    end do FindClosestSpatialTr2Tr


    PRINT *, "Writing spatial distance between training points file..."

    open (unit = 47,  file = OutputFileNameTrMinXYTr, access = "sequential", &
        form = "formatted", action = "write", status = "replace")

    WriteTrSFiles: do i = 1, NumDraws

        WriteTrSpatialTr: do j = 1, DrawSize

            write (unit = 47, fmt = "(I10,I10,I10,F15.5,I10,F15.5)") i, j, &
            TrMinXY2TrainA(j,1,i), TrMinXY2Train(j,1,i), TrMinXY2TrainA(j,2,i),&
            TrMinXY2Train(j,2,i)

        end do WriteTrSpatialTr

    end do WriteTrSFiles

close (unit = 47)


    PRINT *, "Calculating spatial distance between training and validation..."

    ValminusTr = 0.0

    FindClosestSpatialTr2Val: do i = 1, NumDraws

        do j = 1, NumValPoints

                MinXYDist = 1000000.0
                 NearestTrain = 0

                 do l = 1, DrawSize

                 TrainValXDiff = SampleXYTrain (1,l,i) - XYValData (j,1)
                 TrainValYDiff = SampleXYTrain (2,l,i) - XYValData (j,2)
                 TrainValXYDist = SQRT((TrainValXDiff*TrainValXDiff) + &
                       (TrainValYDiff*TrainValYDiff))

                 if (TrainValXYDist .LT. MinXYDist) then

                        MinXYDist = TrainValXYDist
                        NearestTrain = RandomIndices(l,i)

                 end if
```

```
                        end do

                        ValMinXY2Train (j,1,i) = MinXYDist
                        ValMinXY2TrainA (j,1,i) = NearestTrain

                              ValminusTr (:,1) = ValVector(:,j) - &
                                     TrainVector(:,NearestTrain)

                       OnebyOneMatrixB = &
                                     SQRT (MATMUL (TRANSPOSE (ValminusTr), &
                                        ValminusTr))

                        TempValDTr = OnebyOneMatrixB (1,1)

                       ValMinXY2Train (j,2,i) = TempValDTr

                end do

        end do FindClosestSpatialTr2Val


        PRINT *, "Writing spatial distance file between train and val..."

        open (unit = 48,  file = OutputFileNameValMinXYTr, access = "sequential", &
            form = "formatted", action = "write", status = "replace")

        WriteFiles: do i = 1, NumDraws

            WriteValSpatialTr: do j = 1, NumValPoints

                write (unit = 48, fmt = "(I10,I10,F15.5,I10,F15.5)") i, j, &
                ValMinXY2Train(j,1,i), ValMinXY2TrainA(j,1,i), ValMinXY2Train(j,2,i)

            end do WriteValSpatialTr

           end do WriteFiles

    close (unit = 48)


    write (unit = *, fmt = *) "Global Processing Completed"

    write (unit = *, fmt = *) "Remember that draws are separated by two -1s in value files."

end program ksbootsys
```

APPENDIX E: CEARS Library Fortran 90 Code

```
!   Last change:  RHW  22 Mar 2005   11:29 am
!   Background pixel check only enabled for nns, PercentOverallAccuracy, and Recode
!
module CEARSwLib

contains
!
! ComputeNN
!
! Purpose:
!       From input of cluster means and an array of brightness value vectors,
!       find the cluster mean nearest to each brightness value vector in the image
!
! Inputs:
!       Cluster means array
!       Brightness value vector array
!
! Outputs:
!       Nearest neighbor array
!
! Assumptions:
!       Index values in cluster means array are the cluster numbers

subroutine nn (b, k, c, x, m, scx, DistanceImage)

implicit none
        INTEGER (kind = 2), INTENT (IN) :: b
        INTEGER (kind = 4), INTENT (IN) :: k,c
        REAL (kind = 8), INTENT (IN) :: m (b,c)
        REAL (KIND = 8) :: XminusM (b,1)
        REAL (kind = 8) :: MinimumDistance1x1Array (1,1)
        REAL (kind = 8) :: MinimumDistance
        INTEGER (kind = 2) :: ClosestSpectralClass
        INTEGER (KIND = 2), INTENT (INOUT) :: x (b,k)
       INTEGER (KIND = 2), INTENT (OUT) :: scx (k)
        INTEGER (KIND = 4) :: i,j, TempCount
        REAL (KIND = 8), INTENT (OUT) :: DistanceImage (k)

        XminusM = 0.0
        MinimumDistance1x1Array = 0.0
        ClosestSpectralClass = 0
       TempCount = 0
       MinimumDistance = 1000000.0
       scx = 0
       DistanceImage = 0.0

        NumPixelsLoop: do i = 1, k

                ClosestSpectralClass = 0

               IsNotBackground: if ( x (1,i) .NE. 0 ) then

                        TempCount = TempCount + 1

                      NumClustersLoop: do j = 1, c

                              XminusM (:,1) = x(:,i) - m(:,j)
                              MinimumDistance1x1Array = &
                                      MATMUL (TRANSPOSE (XminusM), &
                                        XminusM)

                              if (MinimumDistance1x1Array (1,1) .LT. &
                                      MinimumDistance ) then

                                      MinimumDistance = MinimumDistance1x1Array (1,1)
                                      ClosestSpectralClass = j
                              end if
```

```
                        end DO NumClustersLoop

                          DistanceImage (i) = MinimumDistance
                        MinimumDistance = 1000000.0
                          XminusM = 0.0
                          MinimumDistance1x1Array = 0.0

                end if IsNotBackground

                scx (i) = ClosestSpectralClass
                 ClosestSpectralClass = 0

        end DO NumPixelsLoop

        PRINT *, "TempCount = ", TempCount

end subroutine nn

subroutine Recode (k, scx, sc_ic, icx)

        implicit none

        INTEGER (KIND = 4), INTENT (IN) :: k
        INTEGER (KIND = 2), INTENT (OUT) :: icx (k)
        INTEGER (KIND = 2), INTENT (INOUT) :: scx (k)
        INTEGER (KIND = 4), INTENT (IN) :: sc_ic (:)
        INTEGER (KIND = 4) :: i, TempCount = 0

        icx = 0

        TempCount = 0

        NumPixelsLoop: do i = 1, k

            IsNotBackground: if ( scx (i) .NE. 0 ) then

                TempCount = TempCount + 1

                icx (i) = sc_ic (scx (i))

            end if IsNotBackground

        end DO NumPixelsLoop

end subroutine Recode


subroutine AccPlusAreaEst (k, x, NumPoints, AccPairs, Acres,&
    ErrorMatrix, ForestAreaEstimate, ForestMapMarginal, PercentOverallAccuracy,&
    StandardError, PrecisionFAE, AAPoints)

        implicit none

        INTEGER (kind = 4), INTENT (IN) :: k, NumPoints
        INTEGER (KIND = 2), INTENT (INOUT) :: x (k)
        INTEGER (KIND = 4), INTENT (INOUT) :: AccPairs (NumPoints,2)
        INTEGER (KIND = 4), INTENT (OUT) :: ErrorMatrix (2,2)
        REAL (KIND = 8), INTENT(IN) :: Acres
        REAL (KIND = 8), INTENT (OUT) :: ForestAreaEstimate
        REAL (KIND = 8), INTENT (OUT) :: StandardError
        REAL (KIND = 8), INTENT (OUT) :: ForestMapMarginal
        REAL (KIND = 8), INTENT (OUT) :: PercentOverallAccuracy
        REAL (KIND = 8), INTENT (OUT) :: PrecisionFAE
        INTEGER (KIND = 4), INTENT (OUT) :: AAPoints (NumPoints,2)
        INTEGER (KIND = 4) :: ForestDiv
        INTEGER (KIND = 4) :: NonforestDiv
        REAL (KIND = 8) :: NumPointsReal
        REAL (KIND = 8) :: NumAccurateReal
        REAL (KIND = 8) :: NumNonzeroPixels
        INTEGER (KIND = 4) :: i
```

```
INTEGER (KIND = 4) :: CaseProduct
REAL (KIND = 8) :: NumNonforestPixels
REAL (KIND = 8) :: NumForestPixels
REAL (KIND = 8) :: NonforestMapMarginal
REAL (KIND = 8) :: PfRf
REAL (KIND = 8) :: PnfRf
REAL (KIND = 8) :: FmmxPf
REAL (KIND = 8) :: NfmmxPnf

ErrorMatrix = 0
ForestAreaEstimate = 0.0
 PrecisionFAE = 0.0
StandardError = 0.0
ForestMapMarginal = 0.0
PercentOverallAccuracy = 0.0
PfRf = 0.0
PnfRf = 0.0
FmmxPf = 0.0
NfmmxPnf = 0.0
ForestDiv = 0
NonforestDiv = 0
AAPoints = 0
NumPointsReal = 0.0
NumAccurateReal = 0.0
NumNonzeroPixels = 0.0
CaseProduct = 0
NumNonforestPixels = 0.0
NumForestPixels = 0.0
NonforestMapMarginal = 0.0

PRINT *, "ForestDiv = ", ForestDiv
PRINT *, "NonforestDiv = ", NonforestDiv

ConvertToThreesAndFours: do i = 1, NumPoints


    OneToThree: if ( AccPairs (i,1) .EQ. 1 ) then

        AccPairs (i,1) = 3

    end if OneToThree

    TwoToFour: if (AccPairs (i,1) .EQ. 2) then

        AccPairs (i,1) = 4

    end if TwotoFour

end do ConvertToThreesAndFours

NumPointsLoop: do i = 1, NumPoints

    IsNotBackground: if ( x ( AccPairs (i,2) ) .NE. 0 ) then

        CaseProduct =  x ( AccPairs (i,2) ) * AccPairs (i,1)

          select case (CaseProduct)

            case (3)

                ErrorMatrix (1,1) = ErrorMatrix (1,1) + 1

            case (4)

                ErrorMatrix (2,1) = ErrorMatrix (2,1) + 1

            case (6)

                ErrorMatrix (1,2) = ErrorMatrix (1,2) + 1

            case (8)
```

```
                ErrorMatrix (2,2) = ErrorMatrix (2,2) + 1

         end select

         AAPoints (i,1) = AccPairs (i,2)
         AAPoints (i,2) = CaseProduct

        CaseProduct = 0

    end if IsNotBackground

end DO NumPointsLoop

NumAccurateReal = ErrorMatrix (1,1) + ErrorMatrix (2,2)
NumPointsReal = NumPoints
PercentOverallAccuracy = 100 * NumAccurateReal / NumPointsReal

PRINT *, "NumAccurateReal =  ", NumAccurateReal
PRINT *, "NumPointsReal =  ", NumPointsReal


NumNonzeroPixels = 0.0
NumForestPixels = 0.0
NumNonforestPixels = 0.0

CountForestPixels: do i = 1, k

       if ( x (i) .NE. 0 ) then

       NumNonzeroPixels = NumNonzeroPixels + 1.0

              IfForestPixel: if (x(i) .EQ. 2) then

                     NumForestPixels = NumForestPixels + 1

              end if IfForestPixel

              IfNonforestPixel: if (x(i) .EQ. 1) then

                     NumNonforestPixels = NumNonforestPixels + 1

              end if IfNonforestPixel

end if

end do CountForestPixels

PRINT *, "NumForestPixels =  ", NumForestPixels
PRINT *, "NumNonforestPixels =  ", NumNonforestPixels

ForestMapMarginal = NumForestPixels / (NumForestPixels + NumNonforestPixels)

PRINT *, "ForestMapMarginal =  ", ForestMapMarginal

NonforestMapMarginal = NumNonforestPixels / (NumForestPixels + &
       NumNonforestPixels)

PRINT *, "NonforestMapMarginal =  ", NonforestMapMarginal

PRINT *, "Error Matrix 1,1 = ", ErrorMatrix (1,1)
PRINT *, "Error Matrix 2,1 = ", ErrorMatrix (2,1)
PRINT *, "Error Matrix 1,2 = ", ErrorMatrix (1,2)
PRINT *, "Error Matrix 2,2 = ", ErrorMatrix (2,2)

ForestDiv = ErrorMatrix (1,2) + ErrorMatrix (2,2)
PRINT *, "ForestDiv = ", ForestDiv

NonforestDiv = ErrorMatrix (1,1) + ErrorMatrix (2,1)
PRINT *, "NonforestDiv = ", NonforestDiv
```

```
        if (ForestDiv .NE. 0) then
            PfRf = ErrorMatrix (2,2) / real (ForestDiv, 4)
        end if
        PRINT *, "PfRf = ", PfRf

        if (NonforestDiv .NE. 0) then
            PnfRf = ErrorMatrix (2,1) / real (NonforestDiv, 4)
        end if
        PRINT *, "PnfRf = ", PnfRf

        FmmxPf = ForestMapMarginal * PfRf
        PRINT *, "FmmxPf = ", FmmxPf
        NfmmxPnf = NonforestMapMarginal * PnfRf
        PRINT *, "NfmmxPnf = ", NfmmxPnf

        ForestAreaEstimate = (ForestMapMarginal * PfRf) &
               + (NonforestMapMarginal * PnfRf)
        PRINT *, "ForestAreaEstimate = ", ForestAreaEstimate

        StandardError = (((ForestMapMarginal - FmmxPf) * FmmxPf) / &
                (ForestMapMarginal * NumPoints)) + (((NonforestMapMarginal - NfmmxPnf)*&
                 NfmmxPnf) / (NonforestMapMarginal * NumPoints))
        PRINT *, "StandardError = ", StandardError

        PrecisionFAE = &
               (SQRT(StandardError)*(SQRT(((ForestAreaEstimate)*Acres)/1000000)))
        PRINT *, "PrecisionFAE = ", PrecisionFAE

end subroutine AccPlusAreaEst

! Copy of above added for use when using previously calculated forest map marginal
! with a new error matrix.

subroutine AccPlusAreaEstP2 (k, x, NumPoints, AccPairs, ForestMapMarginal,  &
   ErrorMatrix, ForestAreaEstimate, PercentOverallAccuracy, StandardError, AAPoints)

        implicit none

        INTEGER (kind = 4), INTENT (IN) :: k, NumPoints
        INTEGER (KIND = 2), INTENT (INOUT) :: x (k)
        INTEGER (KIND = 4), INTENT (INOUT) :: AccPairs (NumPoints,2)
        INTEGER (KIND = 4), INTENT (OUT) :: ErrorMatrix (2,2)
        REAL (KIND = 8), INTENT (OUT) :: ForestAreaEstimate
        REAL (KIND = 8), INTENT (OUT) :: StandardError
        REAL (KIND = 8), INTENT (IN) :: ForestMapMarginal
        REAL (KIND = 8), INTENT (OUT) :: PercentOverallAccuracy
        INTEGER (KIND = 4), INTENT (OUT) :: AAPoints (NumPoints,2)
        INTEGER (KIND = 4) :: ForestDiv
        INTEGER (KIND = 4) :: NonforestDiv
        REAL (KIND = 8) :: NumPointsReal
        REAL (KIND = 8) :: NumAccurateReal
        INTEGER (KIND = 4) :: i
        INTEGER (KIND = 4) :: CaseProduct
        REAL (KIND = 8) :: NonforestMapMarginal
        REAL (KIND = 8) :: PfRf
        REAL (KIND = 8) :: PnfRf
        REAL (KIND = 8) :: FmmxPf
        REAL (KIND = 8) :: NfmmxPnf

        ErrorMatrix = 0
        ForestAreaEstimate = 0.0
         StandardError = 0.0
         PercentOverallAccuracy = 0.0
        PfRf = 0.0
        PnfRf = 0.0
        FmmxPf = 0.0
        NfmmxPnf = 0.0
        ForestDiv = 0
        NonforestDiv = 0
        AAPoints = 0
        NumPointsReal = 0.0
```

```
NumAccurateReal = 0.0
CaseProduct = 0
NonforestMapMarginal = 0.0

PRINT *, "ForestDiv = ", ForestDiv
PRINT *, "NonforestDiv = ", NonforestDiv
PRINT *, "Forest map marginal = ", ForestMapMarginal

ConvertToThreesAndFours: do i = 1, NumPoints

    OneToThree: if ( AccPairs (i,1) .EQ. 1 ) then

        AccPairs (i,1) = 3

    end if OneToThree

    TwoToFour: if (AccPairs (i,1) .EQ. 2) then

        AccPairs (i,1) = 4

    end if TwotoFour

end do ConvertToThreesAndFours

NumPointsLoop: do i = 1, NumPoints

    IsNotBackground: if ( x ( AccPairs (i,2) ) .NE. 0 ) then

        CaseProduct =  x ( AccPairs (i,2) ) * AccPairs (i,1)

          select case (CaseProduct)

            case (3)

                ErrorMatrix (1,1) = ErrorMatrix (1,1) + 1

            case (4)

                ErrorMatrix (2,1) = ErrorMatrix (2,1) + 1

            case (6)

                ErrorMatrix (1,2) = ErrorMatrix (1,2) + 1

            case (8)

                ErrorMatrix (2,2) = ErrorMatrix (2,2) + 1

        end select

        AAPoints (i,1) = AccPairs (i,2)
        AAPoints (i,2) = CaseProduct

        CaseProduct = 0

    end if IsNotBackground

end DO NumPointsLoop

NumAccurateReal = ErrorMatrix (1,1) + ErrorMatrix (2,2)
NumPointsReal = NumPoints
PercentOverallAccuracy = 100 * NumAccurateReal / NumPointsReal

PRINT *, "NumAccurateReal =  ", NumAccurateReal
PRINT *, "NumPointsReal =  ", NumPointsReal
PRINT *, "PercentOverallAccuracy = ", PercentOverallAccuracy

NonforestMapMarginal = (1 - ForestMapMarginal)
PRINT *, "ForestMapMarginal =  ", ForestMapMarginal
PRINT *, "NonforestMapMarginal =  ", NonforestMapMarginal
```

```
        PRINT *, "Error Matrix 1,1 = ", ErrorMatrix (1,1)
        PRINT *, "Error Matrix 2,1 = ", ErrorMatrix (2,1)
        PRINT *, "Error Matrix 1,2 = ", ErrorMatrix (1,2)
        PRINT *, "Error Matrix 2,2 = ", ErrorMatrix (2,2)

        ForestDiv = ErrorMatrix (1,2) + ErrorMatrix (2,2)
        PRINT *, "ForestDiv = ", ForestDiv
        NonforestDiv = ErrorMatrix (1,1) + ErrorMatrix (2,1)
        PRINT *, "NonforestDiv = ", NonforestDiv

        if (ForestDiv .NE. 0) then
            PfRf = ErrorMatrix (2,2) / real (ForestDiv, 4)
        end if
        PRINT *, "PfRf = ", PfRf

        if (NonforestDiv .NE. 0) then
            PnfRf = ErrorMatrix (2,1) / real (NonforestDiv, 4)
        end if
        PRINT *, "PnfRf = ", PnfRf

        FmmxPf = ForestMapMarginal * PfRf
        PRINT *, "FmmxPf = ", FmmxPf
        NfmmxPnf = NonforestMapMarginal * PnfRf
        PRINT *, "NfmmxPnf = ", NfmmxPnf

        ForestAreaEstimate = (ForestMapMarginal * PfRf) &
               + (NonforestMapMarginal * PnfRf)
        PRINT *, "ForestAreaEstimate = ", ForestAreaEstimate

        StandardError = (((ForestMapMarginal - FmmxPf) * FmmxPf) / &
                (ForestMapMarginal * NumPoints)) + (((NonforestMapMarginal - NfmmxPnf)*&
                 NfmmxPnf) / (NonforestMapMarginal * NumPoints))
        PRINT *, "StandardError = ", StandardError

end subroutine AccPlusAreaEstP2

subroutine ComputeAccuracyOnly (k, x, NumPoints, AccPairs, ErrorMatrix, &
    PercentOverallAccuracy, AAPoints)

        implicit none

        INTEGER (kind = 4), INTENT (IN) :: k, NumPoints
        INTEGER (KIND = 2), INTENT (INOUT) :: x (k)
        INTEGER (KIND = 4), INTENT (INOUT) :: AccPairs (NumPoints,2)
        INTEGER (KIND = 4), INTENT (OUT) :: ErrorMatrix (2,2)
        REAL (KIND = 8), INTENT (OUT) :: PercentOverallAccuracy
        REAL (KIND = 8) :: NumPointsReal
        REAL (KIND = 8) :: NumAccurateReal
        REAL (KIND = 8) :: NumNonzeroPixels
        INTEGER (KIND = 4) :: i
        INTEGER (KIND = 4) :: CaseProduct
        INTEGER (KIND = 4), INTENT (OUT) :: AAPoints (NumPoints,2)

        ErrorMatrix = 0
        PercentOverallAccuracy = 0.0
        AAPoints = 0
        NumPointsReal = 0.0
        NumAccurateReal = 0.0
        NumNonzeroPixels = 0.0
        CaseProduct = 0

        ConvertToThreesAndFours: do i = 1, NumPoints

            OneToThree: if ( AccPairs (i,1) .EQ. 1 ) then

                AccPairs (i,1) = 3

            end if OneToThree

            TwoToFour: if (AccPairs (i,1) .EQ. 2) then
```

```
                AccPairs (i,1) = 4

            end if TwotoFour

        end do ConvertToThreesAndFours

        NumPointsLoop: do i = 1, NumPoints

            IsNotBackground: if ( x (i) .NE. 0 ) then

                CaseProduct =  x (i) * AccPairs (i,1)

                  select case (CaseProduct)

                    case (3)

                        ErrorMatrix (1,1) = ErrorMatrix (1,1) + 1

                    case (4)

                        ErrorMatrix (2,1) = ErrorMatrix (2,1) + 1

                    case (6)

                        ErrorMatrix (1,2) = ErrorMatrix (1,2) + 1

                    case (8)

                        ErrorMatrix (2,2) = ErrorMatrix (2,2) + 1

                end select

                AAPoints (i,1) = AccPairs (i,2)
                AAPoints (i,2) = CaseProduct

                CaseProduct = 0

            end if IsNotBackground

        end DO NumPointsLoop

        NumAccurateReal = ErrorMatrix (1,1) + ErrorMatrix (2,2)
        NumPointsReal = NumPoints
        PercentOverallAccuracy = 100 * NumAccurateReal / NumPointsReal

        PRINT *, "NumAccurateReal =  ", NumAccurateReal
        PRINT *, "NumPointsReal =  ", NumPointsReal
        PRINT *, "Percent Overall Accuracy = ", PercentOverallAccuracy

end subroutine ComputeAccuracyOnly

subroutine ComputeMcRobertsAreaEstimate (k, x, NumPoints, AccPairs, ForestMapMarginal, &
        PlotProportionForest, Acres, ForestCount, NonforestCount, MeanPlotPropForest, &
        MeanPlotPropNonforest, WeightedProportion, ForestArea, ForestStratumVariance, &
        NonforestStratumVariance, VariancesSum, VarianceForestArea, FAEPrecision)

        INTEGER (kind = 4), INTENT (IN) :: k, NumPoints
        INTEGER (KIND = 2), INTENT (INOUT) :: x (k)
        INTEGER (KIND = 4), INTENT (INOUT) :: AccPairs (NumPoints,2)
        REAL (KIND = 8), INTENT (IN) :: ForestMapMarginal
        REAL (KIND = 8), INTENT (IN) :: PlotProportionForest (NumPoints)
        REAL (KIND = 8), INTENT (IN) :: Acres
         INTEGER (kind = 4), INTENT (IN) :: ForestCount
        INTEGER (kind = 4), INTENT (IN) :: NonforestCount
        INTEGER (KIND = 4) :: i, j
        INTEGER (KIND = 4) :: FCount, NfCount
        REAL (KIND = 8) :: ForestCountReal
         REAL (KIND = 8) :: NonforestCountReal
        REAL (KIND = 8) :: NonforestMapMarginal
        REAL (KIND = 8) :: PlotsInForest (ForestCount)
        REAL (KIND = 8) :: PlotsInNonforest (NonforestCount)
```

```
REAL (KIND = 8) :: ForestSumPlotProp
REAL (KIND = 8) :: NonforestSumPlotProp
REAL (KIND = 8), INTENT (OUT) :: MeanPlotPropForest
REAL (KIND = 8), INTENT (OUT) :: MeanPlotPropNonforest
REAL (KIND = 8), INTENT (OUT) :: WeightedProportion
REAL (KIND = 8), INTENT (OUT) :: ForestArea
REAL (KIND = 8) :: SqDiffPlotsInForest (ForestCount)
REAL (KIND = 8) :: SqDiffPlotsInNonforest (NonforestCount)
REAL (KIND = 8) :: ForestSumSqDiff
REAL (KIND = 8) :: NonforestSumSqDiff
REAL (KIND = 8), INTENT (OUT) :: ForestStratumVariance
REAL (KIND = 8), INTENT (OUT) :: NonforestStratumVariance
REAL (KIND = 8), INTENT (OUT) :: VariancesSum
REAL (KIND = 8), INTENT (OUT) :: VarianceForestArea
REAL (KIND = 8), INTENT (OUT) :: FAEPrecision

FCount = 0
NfCount = 0
ForestCountReal = 0.0
NonforestCountReal = 0.0
WeightedProportion = 0.0
ForestArea = 0.0
VariancesSum = 0.0
VarianceForestArea = 0.0
FAEPrecision = 0.0
PlotsInForest = 0.0
PlotsInNonforest =0.0
SqDiffPlotsInForest = 0.0
SqDiffPlotsInNonforest = 0.0
NonforestMapMarginal = 0.0
ForestSumPlotProp = 0.0
NonforestSumPlotProp = 0.0
MeanPlotPropForest = 0.0
MeanPlotPropNonforest = 0.0
ForestSumSqDiff = 0.0
NonforestSumSqDiff = 0.0
ForestStratumVariance = 0.0
NonforestStratumVariance = 0.0

NonforestMapMarginal = (1 - ForestMapMarginal)

PlotsByClass: do j = 1, NumPoints

       if ( x (AccPairs (j,2)) .NE. 0 ) then

              IfForestPixel: if (x (AccPairs (j,2)) .EQ. 2) then

                       FCount = FCount + 1
                     PlotsInForest (FCount) = PlotProportionForest (j)


              end if IfForestPixel

              IfNonforestPixel: if (x (AccPairs (j,2)) .EQ. 1) then

                       NfCount = NfCount + 1
                     PlotsInNonforest (NfCount) = PlotProportionForest (j)

              end if IfNonforestPixel

       end if

end do PlotsByClass

PRINT *, "FCount = ", FCount
PRINT *, "NfCount = ", NfCount

         do i = 1, ForestCount

              ForestSumPlotProp = ForestSumPlotProp + PlotsInForest (i)
```

```
          end do

          do i = 1, NonforestCount

               NonforestSumPlotProp = NonforestSumPlotProp + PlotsInNonforest (i)

          end do

PRINT *, "ForestSumPlotProp = ", ForestSumPlotProp
PRINT *, "NonforestSumPlotProp = ", NonforestSumPlotProp

ForestCountReal = ForestCount
NonforestCountReal = NonforestCount

MeanPlotPropForest = ForestSumPlotProp / ForestCountReal
MeanPlotPropNonforest = NonforestSumPlotProp / NonforestCountReal

PRINT *, "MeanPlotPropForest = ", MeanPlotPropForest
PRINT *, "MeanPlotPropNonforest = ", MeanPlotPropNonforest

WeightedProportion = ((ForestMapMarginal * MeanPlotPropForest) + &
        (NonforestMapMarginal * MeanPlotPropNonforest))

PRINT *, "WeightedProportion = ", WeightedProportion

ForestArea = Acres * WeightedProportion

PRINT *, "ForestArea = ", ForestArea

  do i = 1, ForestCount

  SqDiffPlotsInForest (i) = ((PlotsInForest (i) - MeanPlotPropForest) * &
        (PlotsInForest (i) - MeanPlotPropForest))

   end do

  do i = 1, NonforestCount

  SqDiffPlotsInNonforest (i) = ((PlotsInNonforest (i) - &
              MeanPlotPropNonforest) * &
        (PlotsInNonforest (i) - MeanPlotPropNonforest))

   end do

          do i = 1, ForestCount

               ForestSumSqDiff = ForestSumSqDiff + SqDiffPlotsInForest (i)

          end do

          do i = 1, NonforestCount

               NonforestSumSqDiff = NonforestSumSqDiff + SqDiffPlotsInNonforest (i)

          end do

   PRINT *, "ForestSumSqDiff = ", ForestSumSqDiff
   PRINT *, "NonforestSumSqDiff = ", NonforestSumSqDiff

   ForestStratumVariance = ((1 / (ForestCountReal - 1)) * ForestSumSqDiff)
   NonforestStratumVariance = ((1 / (NonforestCountReal - 1)) * &
        NonforestSumSqDiff)

   PRINT *, "Forest Map Marginal = ", ForestMapMarginal
   PRINT *, "Nonforest Map Marginal = ", NonforestMapMarginal
   PRINT *, "Forest Stratum Variance = ", ForestStratumVariance
   PRINT *, "Nonforest Stratum Variance = ", NonforestStratumVariance
   PRINT *, "Forest Count Real = ", ForestCountReal
   PRINT *, "Nonforest Count Real = ", NonforestCountReal

   VariancesSum = (((ForestMapMarginal * ForestMapMarginal) * &
```

```
                  ForestStratumVariance) / ForestCountReal) + &
                 (((NonforestMapMarginal * NonforestMapMarginal) *  &
                NonforestStratumVariance) /NonforestCountReal)

          VarianceForestArea = (Acres * Acres) * VariancesSum

          FAEPrecision = ((SQRT(VarianceForestArea)) / ForestArea) * &
               SQRT(ForestArea / 1000000)

          PRINT *, "Acres = ", Acres
          PRINT *, "ForestArea = ", ForestArea
          PRINT *, "VariancesSum = ", VariancesSum
          PRINT *, "VarianceForestArea = ", VarianceForestArea
          PRINT *, " FAEPrecision = ", FAEPrecision

end subroutine ComputeMcRobertsAreaEstimate

    recursive subroutine CompareTestnNumDraws(Indices, NumDraws)

        INTEGER (KIND = 4), INTENT (IN) :: NumDraws
        INTEGER (KIND = 4), INTENT (INOUT) :: Indices (NumDraws)
        INTEGER (KIND = 4) :: i
        REAL (KIND = 8) :: SingleRandomNumber

        CompareIndices: do i = 1, NumDraws

               if (Indices (i) .NE. i) then

                Indices (i) = Indices (i)

                else

               CALL random_number (SingleRandomNumber)
               Indices (i) = int (SingleRandomNumber * NumDraws + 1, 4)

                call CompareTestnNumDraws(Indices,NumDraws)

                end if

        end do CompareIndices

     end subroutine CompareTestnNumDraws


end module CEARSwLib
```

APPENDIX F: Quick Sort Fortran 90 Code

```
module quicksortw_mod
  implicit none

  logical :: UseRandom  !determines whether select/median find will be
  !randomized

  contains

    !Partition will partition an array for a quicksort by using the first value
    !in the array as the pivot.  Partition will return an index, and all
    !elements to the left and including that index will be less than the pivot,
    !and all elements to the right of that index will be greater than the pivot
    !value.

    function Partition(A,p,r)
      implicit none
      integer, intent(inout), dimension(:,:) :: A  !input array
      integer, intent(in) :: p  !lower index of subarray to be
      !partitioned
      integer, intent(in) :: r  !upper index of subarray to be
      !partitioned
      integer :: Partition      !all elements having an index below
      !and including this index will be less than all elements having an index
      !greater than this index

      !------------------------------------------------------------------------
      !variable declarations
      !------------------------------------------------------------------------
      integer :: i,      &  !used to iterate starting at lower bound
           j,      &  !used to iterate starting at upper bound
           pivot,  &  !holds value of pivot
           temp, &    !used to exchange values in two array locations
            temploc
      !------------------------------------------------------------------------

      !store the pivot value
      pivot = A(2,p)

      !initialize i and j
      i = p - 1
      j = r + 1

      do
         !loop until there is an element at end of array with value lower than
         !the pivot
         JLoop: do
            j = j - 1
            if (A(2,j).le.pivot) then
               exit JLoop
            end if
         end do JLoop

         !loop until there is an element at beginning of array with value
         !greater than the pivot
         ILoop: do
            i = i + 1
            if (A(2,i).ge.pivot) then
               exit ILoop
            end if
         end do ILoop

         if (i.lt.j) then
            !exchange values in positions i and j
            temp = A(2,j)
            temploc = A(1,j)
            A(2,j) = A(2,i)
            A(1,j) = A(1,i)
            A(2,i) = temp
```

```
        A(1,i) = temploc
      else
         !return j
         Partition = j
         return
      end if
   end do
end function Partition

!The purpose of RandomPartition is to randomly select the pivot for
!partition to minimize the selection of bad pivot values.  RANDOM_SEED
!should be called once before this call to initialize the random number
!generator.  Note: p should always be less than or equal to r.
function RandomPartition(A,p,r)
   implicit none
   integer, intent(inout), dimension(:,:) :: A  !array to be
   !partitioned
   integer, intent(in) :: p  !lower bound of subarray to be
   !partitioned
   integer, intent(in) :: r  !upper bound of subarray to be
   !partitioned
   integer :: RandomPartition  !return index such that all elements
   !with an index less than or equal to this have a value less than or equal
   !to all elements with an index greater to the return index

   !-------------------------------------------------------------------------
   !variable declarations
   !-------------------------------------------------------------------------
   integer :: index !random index
   integer :: temp  !used to exchange array values
   integer :: temploc  !used to exchange array values
   real :: random   !random number between 0 and 1
   !-------------------------------------------------------------------------

   call RANDOM_NUMBER(random)

   index = random * (r-p)
   index = p + index

   !testing purposes:
   if (index .gt. r) then
      print *, "Index is greater than upper bound"
      return
   end if
   !exchange values
   temp = A(2,index)
   temploc = A(1,index)
   A(2,index) = A(2,p)
   A(1,index) = A(1,p)
   A(2,p) = temp
   A(1,p) = temploc
   !debugging info
   !print *, "initial array (after random) is ", A
   RandomPartition = Partition(A,p,r)
   return
end function RandomPartition

recursive subroutine QuickSort(A,left, right)
   implicit none
   integer, intent(inout), dimension(:,:) :: A  !array to be sorted
   integer, intent(in) :: left  !left index
   integer, intent(in) :: right !right index

   !-------------------------------------------------------------------------
   !variable declarations
   !-------------------------------------------------------------------------
   integer :: I  !index returned by Partition
   !-------------------------------------------------------------------------

   if (left < right) then
      if (UseRandom) then
```

```
            I = Partition(A, left, right)
         else
            I = RandomPartition(A, left, right)
         end if
         call QuickSort(A, left, I)
         call QuickSort(A, I+1, right)
      end if

   end subroutine QuickSort

end module quicksortw_mod
```

APPENDIX G: Euclidian Distance and Cross-Validation Fortran 90 Code

```
!     Last change:  RHW  30 Jun 2005    0:18 am
program EucDistCrossValidationAcc

        implicit none

        INTERFACE

            function ComputeBandMask (ClassNumber, NumBands) result (BandMask)

                implicit none

                INTEGER (KIND = 4), INTENT (IN) :: ClassNumber
                INTEGER (KIND = 2), INTENT (IN) :: NumBands
                INTEGER (KIND = 2) :: BandMask(NumBands)
                INTEGER (KIND = 8) :: Quotient
                INTEGER (KIND = 8) :: Remainder
                INTEGER (KIND = 8) :: i

            end function ComputeBandMask

       END INTERFACE


        INTEGER (KIND = 4), PARAMETER :: NumTotalSamples = 165000
        INTEGER (KIND = 2), PARAMETER :: b = 6
        INTEGER (KIND = 2), PARAMETER :: NumInfoClasses = 2
        INTEGER (KIND = 2), PARAMETER :: MaxSampleSize = 500
        INTEGER (KIND = 2), PARAMETER :: MaxNumDraws = 800
        INTEGER (KIND = 2), PARAMETER :: S_LEN = 32
        INTEGER (KIND = 2), PARAMETER :: NumDrawsPerSample = 100
        INTEGER (KIND = 4), PARAMETER :: NumSets = 2**b

        type DrawValues
            INTEGER (KIND = 2) :: x (b,MaxSampleSize)
            INTEGER (KIND = 2) :: InfoClass (MaxSampleSize)
            INTEGER (KIND = 2) :: CVInfoClass (MaxSampleSize)
            REAL (KIND = 4) :: CrossValidationAccuracy(NumSets)
            REAL (KIND = 4) :: MinimumEuclideanDistance(NumSets)
            REAL (KIND = 4) :: AverageEuclideanDistance(NumSets)
            INTEGER (KIND = 2) :: BandMasks(b,NumSets)
            INTEGER (KIND = 2) :: SampleSize
            INTEGER (KIND = 2) :: DrawMult
        end type DrawValues

        type (DrawValues) :: Draw (MaxNumDraws)

        INTEGER (KIND = 4) :: g,h,i,j ! Loop counters
        INTEGER (KIND = 2) :: CurrentDrawNumber, LastDrawNumber
        INTEGER (KIND = 2) :: CurrentSampleNumber
        INTEGER (KIND = 2) :: NumDraws
        INTEGER (KIND = 2) :: CurrClass
        INTEGER (KIND = 2) :: DrawMultiplier

        INTEGER (KIND = 2) :: BandMask (b)

        REAL (KIND = 4) :: NumAccurate
        REAL (KIND = 4) :: MinDistance
        REAL (KIND = 4) :: MinDistanceBetweenInfoClasses
        REAL (KIND = 4) :: SumDistance
        REAL (KIND = 4) :: SumNumber
        REAL (KIND = 4) :: XminusM (b,1)
        REAL (kind = 4) :: Distance1x1Array (1,1)

        CHARACTER (LEN = S_LEN) :: InputFileNameB1
        CHARACTER (LEN = S_LEN) :: InputFileNameB2
        CHARACTER (LEN = S_LEN) :: InputFileNameB3
        CHARACTER (LEN = S_LEN) :: InputFileNameB4
        CHARACTER (LEN = S_LEN) :: InputFileNameB5
```

```
CHARACTER (LEN = S_LEN) :: InputFileNameB6
CHARACTER (LEN = S_LEN) :: InputFileNameDrawNumber
CHARACTER (LEN = S_LEN) :: InputFileNameInfoClass
CHARACTER (LEN = S_LEN) :: InputFileNameSampleSize
CHARACTER (LEN = S_LEN) :: ParametersFileName
CHARACTER (LEN = S_LEN) :: OutputFileName

write (unit = *, FMT = *) "Name of parameters file: "
read (unit = *, FMT = *) ParametersFileName

       open (unit = 10, file=ParametersFileName, status = "old", action = "read")
       read (unit = 10, fmt = *) InputFileNameB1
       open (unit = 10, file=ParametersFileName, status = "old", action = "read")
       read (unit = 10, fmt = *) InputFileNameB2
       open (unit = 10, file=ParametersFileName, status = "old", action = "read")
       read (unit = 10, fmt = *) InputFileNameB3
       open (unit = 10, file=ParametersFileName, status = "old", action = "read")
       read (unit = 10, fmt = *) InputFileNameB4
       open (unit = 10, file=ParametersFileName, status = "old", action = "read")
       read (unit = 10, fmt = *) InputFileNameB5
       open (unit = 10, file=ParametersFileName, status = "old", action = "read")
       read (unit = 10, fmt = *) InputFileNameB6
       open (unit = 10, file=ParametersFileName, status = "old", action = "read")
       read (unit = 10, fmt = *) InputFileNameDrawNumber
       open (unit = 10, file=ParametersFileName, status = "old", action = "read")
       read (unit = 10, fmt = *) InputFileNameSampleSize
       open (unit = 10, file=ParametersFileName, status = "old", action = "read")
       read (unit = 10, fmt = *) InputFileNameInfoClass
       open (unit = 10, file=ParametersFileName, status = "old", action = "read")
       read (unit = 10, fmt = *) OutputFileName

write (unit = *, FMT = *) "Input File Names: "
write (unit = *, FMT = *) "    ", InputFileNameB1, " (Unit = 11)"
write (unit = *, FMT = *) "    ", InputFileNameB2, " (Unit = 12)"
write (unit = *, FMT = *) "    ", InputFileNameB3, " (Unit = 13)"
write (unit = *, FMT = *) "    ", InputFileNameB4, " (Unit = 14)"
write (unit = *, FMT = *) "    ", InputFileNameB5, " (Unit = 15)"
write (unit = *, FMT = *) "    ", InputFileNameB6, " (Unit = 16)"
write (unit = *, FMT = *) "    ", InputFileNameDrawNumber, " (Unit = 17)"
write (unit = *, FMT = *) "    ", InputFileNameSampleSize, " (Unit = 18)"
write (unit = *, FMT = *) "    ", InputFileNameInfoClass, " (Unit = 19)"
write (unit = *, FMT = *) "    ", "parameters.in (Unit = 10)"
write (unit = *, FMT = *) "Number of Bands: "
write (unit = *, FMT = *) "    ", b

write (unit = *, FMT = *) "Number of Information Classes: "
write (unit = *, FMT = *) "    ", NumInfoClasses

write (unit = *, FMT = *) "Number of Total Samples: "
write (unit = *, FMT = *) "    ", NumTotalSamples

write (unit = *, FMT = *) "Output File Names: "
write (unit = *, FMT = *) "    ", OutputFileName, " (Unit = 2)"

write (unit = *, fmt = *) "Reading input files into memory..."
open (unit = 11, file = InputFileNameB1, &
       form = "formatted", status = "old", action = "read")
open (unit = 12, file = InputFileNameB2, &
       form = "formatted", status = "old", action = "read")
open (unit = 13, file = InputFileNameB3, &
       form = "formatted", status = "old", action = "read")
open (unit = 14, file = InputFileNameB4, &
       form = "formatted", status = "old", action = "read")
open (unit = 15, file = InputFileNameB5, &
       form = "formatted", status = "old", action = "read")
open (unit = 16, file = InputFileNameB6, &
       form = "formatted", status = "old", action = "read")
open (unit = 17, file = InputFileNameDrawNumber, &
       form = "formatted", status = "old", action = "read", ACCESS="sequential")
open (unit = 18, file = InputFileNameSampleSize, &
       form = "formatted", status = "old", action = "read")
```

```
        open (unit = 19, file = InputFileNameInfoClass, &
               form = "formatted", status = "old", action = "read")

        CurrentSampleNumber = 0
        CurrentDrawNumber = 1
        DrawMultiplier = 0

!       PRINT *, "CurrentDrawNumber = ", CurrentDrawNumber

!        PRINT *, "NumTotalSamples = ", NumTotalSamples
        ReadDataLoop: do i = 1, NumTotalSamples

            if ( (modulo(CurrentDrawNumber,int(NumDrawsPerSample,2)) .EQ. 0) .AND. &
                    (CurrentSampleNumber .EQ. Draw(CurrentDrawNumber)%SampleSize) ) then

                DrawMultiplier = DrawMultiplier + 1
!                PRINT *, "DrawMultiplier = ", DrawMultiplier

            endif


            Draw(CurrentDrawNumber)%DrawMult = DrawMultiplier

!            PRINT *, "i = ", i
              LastDrawNumber = CurrentDrawNumber
!            PRINT *, "LastDrawNumber = ", LastDrawNumber
              read (unit = 17, fmt = "(I5)") CurrentDrawNumber


              CurrentDrawNumber = (DrawMultiplier * NumDrawsPerSample) + &
                           CurrentDrawNumber
!                     PRINT *, "CurrentDrawNumber = ",CurrentDrawNumber
            read (unit = 18, fmt = "(I4)") Draw(CurrentDrawNumber)%SampleSize
!            PRINT *, "Draw(CurrentDrawNumber)%SampleSize", &
                      Draw(CurrentDrawNumber)%SampleSize

               if (CurrentDrawNumber .EQ. LastDrawNumber) then

                       CurrentSampleNumber = CurrentSampleNumber + 1
!                       PRINT *, "CurrentSampleNumber = ", CurrentSampleNumber

               else
                       CurrentSampleNumber = 1
!                       PRINT *, "CurrentSampleNumber = ", CurrentSampleNumber

               end if

               read (unit = 11, fmt = "(I4)") Draw(CurrentDrawNumber) &
                      %x (1,CurrentSampleNumber)
              read (unit = 12, fmt = "(I4)") Draw(CurrentDrawNumber) &
                      %x (2,CurrentSampleNumber)
              read (unit = 13, fmt = "(I4)") Draw(CurrentDrawNumber) &
                      %x (3,CurrentSampleNumber)
              read (unit = 14, fmt = "(I4)") Draw(CurrentDrawNumber) &
                      %x (4,CurrentSampleNumber)
              read (unit = 15, fmt = "(I4)") Draw(CurrentDrawNumber) &
                      %x (5,CurrentSampleNumber)
              read (unit = 16, fmt = "(I4)") Draw(CurrentDrawNumber) &
                      %x (6,CurrentSampleNumber)
              read (unit = 19, fmt = "(I2)") Draw(CurrentDrawNumber) &
                      %InfoClass (CurrentSampleNumber)
!               PRINT *, Draw(CurrentDrawNumber)%x (1,CurrentSampleNumber), &
!                   Draw(CurrentDrawNumber)%x (2,CurrentSampleNumber), &
!                   Draw(CurrentDrawNumber)%x (3,CurrentSampleNumber), &
!                   Draw(CurrentDrawNumber)%x (4,CurrentSampleNumber), &
!                   Draw(CurrentDrawNumber)%x (5,CurrentSampleNumber), &
!                   Draw(CurrentDrawNumber)%x (6,CurrentSampleNumber), &
!                   Draw(CurrentDrawNumber)%InfoClass (CurrentSampleNumber)
!               PAUSE

        end do ReadDataLoop
```

```
        NumDraws = CurrentDrawNumber
 !       PRINT *, "NumDraws = ", NumDraws

        close (unit = 11)
        close (unit = 12)
        close (unit = 13)
        close (unit = 14)
        close (unit = 15)
        close (unit = 16)
        close (unit = 17)
        close (unit = 18)
        close (unit = 19)

        PRINT *, "Files read successfully"

 !       do i = 1, NumDraws

 !               PRINT *, Draw(i)%SampleSize

 !       end do

               BandMask = 0
        MinDistance = 10000.
        MinDistanceBetweenInfoClasses = 10000.
        SumDistance = 0.
        SumNumber = 0.
        NumAccurate = 0.

!       PRINT *, "NumDraws = ", NumDraws

            BandCombinationLoop: do g = 2, NumSets

             PRINT *, "g = ", g

             BandMask = ComputeBandMask (g, b)

             NumDrawsLoopCV: do h = 1, NumDraws

                 Draw(h)%BandMasks(:,g) = BandMask

    !            PRINT *, "In NumDrawsLoopCV, h = ", h

                    NumSamplesLoopOneCV: do i = 1, Draw(h)%SampleSize

    !                PRINT *, "In NumSamplesLoopOneCV, i = ", i
    !                   PRINT *, "Draw(h)%SampleSize = ", Draw(h)%SampleSize

                           NumSamplesLoopTwoCV: do j = 1, Draw(h)%SampleSize

    !                    PRINT *, "In NumSamplesLoopTwoCV, j = ", j

                                 LeaveOneOut: if (j .NE. i) then

    !                               PRINT *, "In LeaveOneOut (i .NE. j)"
                                  XMinusM(:,1) = BandMask * (Draw(h)%x(:,i)- &
                                               Draw(h)%x(:,j))
    !                           PRINT *, "Draw(h)%x(:,i) = ", Draw(h)%x(:,i)
    !                           PRINT *, "Draw(h)%x(:,j) = ", Draw(h)%x(:,j)
!                               PRINT *, "XMinusM(:,1) = ", XMinusM(:,1)
!                               PAUSE

                                Distance1x1Array = &
                                   sqrt(MATMUL (TRANSPOSE (XminusM), XminusM))

    !                             PRINT *, "Distance1x1Array = ", Distance1x1Array

                                InfoClassesDifferent: if ( Draw(h)%InfoClass(i) .NE. &
                                               Draw(h)%InfoClass(j) ) then

    !                               PRINT *, "InfoclassesDifferent"
```

```
                                    SumDistance = SumDistance + Distance1x1Array(1,1)
                                    SumNumber = SumNumber + 1.
!                                  PRINT *, "SumDistance & SumNumber = ", SumDistance, &
                                           SumNumber

                                    BestMinDistED: if (Distance1x1Array (1,1) < &
                                           MinDistanceBetweenInfoClasses) then

!                                     PRINT *, "BestMinDistED, (Distance1x1Array (1,1) < &
!                                            MinDistanceBetweenInfoClasses)"
                                     MinDistanceBetweenInfoClasses = Distance1x1Array(1,1)
!                                       PRINT *, "MinDistanceBetweenInfoClasses = ", &
!                                                MinDistanceBetweenInfoClasses
!                                       PAUSE

                                    END if BestMinDistED

                                END if InfoClassesDifferent

                                     BestMinDistCV: if (Distance1x1Array (1,1) < &
                                        MinDistance) then

!                                    PRINT *, "BestMinDistCV, (Distance1x1Array (1,1) < &
!                                           MinDistance)"
                                    MinDistance = Distance1x1Array(1,1)
!                                    PRINT *, MinDistance
                                    CurrClass = Draw(h)%InfoClass(j)
!                                    PRINT *, "CurrClass = ", CurrClass
                                END if BestMinDistCV

                                  end if LeaveOneOut

!                           PAUSE

                                end DO NumSamplesLoopTwoCV
!                    PRINT *, "Exited NumSamplesLoopTwoCV"

!                    PRINT *, "CurrClass at end of loop = ", CurrClass
                    Draw(h)%CVInfoClass(i) = CurrClass
                    if ( Draw(h)%CVInfoClass(i) .EQ. Draw(h)%InfoClass(i) ) then

                        NumAccurate = NumAccurate + 1.
!                        PRINT *, "NumAccurate = ", NumAccurate

                    end if
                    MinDistance = 10000.

!                    PAUSE

                    END do NumSamplesLoopOneCV
!                    PRINT *, "Exited NumSamplesLoopOneCV"

!                    PRINT *, "h = ", h
!                    PRINT *, "Draw(h)%SampleSize ",  Draw(h)%SampleSize
                    Draw(h)%MinimumEuclideanDistance(g) = MinDistanceBetweenInfoClasses
!                    PRINT *, Draw(h)%MinimumEuclideanDistance(g)
                    Draw(h)%AverageEuclideanDistance(g) = SumDistance / SumNumber
                    Draw(h)%CrossValidationAccuracy(g) = NumAccurate / &
                              real(Draw(h)%SampleSize, 4)
!                    PRINT *, NumAccurate, real(Draw(h)%SampleSize, 4), &
                              Draw(h)%CrossValidationAccuracy(g)

                    MinDistanceBetweenInfoClasses = 10000.
                    NumAccurate = 0.
                    SumDistance = 0.
                    SumNumber = 0.

!                    PAUSE

                END do NumDrawsLoopCV
!                PRINT *, "ExitedNumDrawsLoopCV"
```

```
        END do BandCombinationLoop

            open (unit = 2, file = OutputFileName, &
               form = "formatted", status = "replace", action = "write")

        PRINT *, "Writing output file..."

        write (unit = 2, fmt = *) "Draw SampleSize BandMask Min Avg Acc"

            WriteDrawLoop: do h = 1, NumDraws
                WriteNumSetsLoop: do g = 2, NumSets

                 if (h .GT. NumDrawsPerSample) then

                     CurrentDrawNumber = modulo(h,Draw(h)%DrawMult * NumDrawsPerSample)
!                     PRINT *, "CurrentDrawNumber = ", CurrentDrawNumber

                 else

                     CurrentDrawNumber = h

                 end if

                 write (unit = 2, fmt = "(I5,I4,I2,I2,I2,I2,I2,I2,F10.4,F10.4,F10.6)") &

                     CurrentDrawNumber, &
                     Draw(h)%SampleSize, &
                     Draw(h)%BandMasks(1,g), &
                     Draw(h)%BandMasks(2,g), &
                     Draw(h)%BandMasks(3,g), &
                     Draw(h)%BandMasks(4,g), &
                     Draw(h)%BandMasks(5,g), &
                     Draw(h)%BandMasks(6,g), &
                     Draw(h)%MinimumEuclideanDistance(g), &
                     Draw(h)%AverageEuclideanDistance(g), &
                     Draw(h)%CrossValidationAccuracy(g)

                end do WriteNumSetsLoop

               end do WriteDrawLoop

        close (unit = 2)

        PRINT *, "Output file successfully written and closed."

end program EucDistCrossValidationAcc

function ComputeBandMask (ClassNumber, NumBands) result (BandMask)

    implicit none

    INTEGER (KIND = 4), INTENT (IN) :: ClassNumber
    INTEGER (KIND = 2), INTENT (IN) :: NumBands
    INTEGER (KIND = 2) :: BandMask(NumBands)
    INTEGER (KIND = 4) :: Quotient
    INTEGER (KIND = 4) :: Remainder
    INTEGER (KIND = 4) :: i

    BandMask = 0
    Quotient = ClassNumber - 1

    do i = 1, NumBands

 !           PRINT *, "i = ", i
 !           PRINT *, "Quotient = ", Quotient

            Remainder = modulo (Quotient,int(2,4))

 !           PRINT *, "Remainder = ", Remainder
```

```
            BandMask (NumBands - i + 1) = Remainder
 !           PRINT *, "BandMask = ", BandMask
            Quotient = Quotient / 2
    end do
end function ComputeBandMask
```