Application & Evaluation of Composite Estimator on Forest Inventory Data with ACAS Software Development

Murray Todd Williams

December 12, 2003

Abstract

Techniques in estimating large-scale forest inventories often require some assessment of misclassification errors between different measurement systems. The Composite Estimator, a statistical technique based on the Kalman Filter and developed by Dr. Ray Czaplewski, provides a means of estimating misclassifiation errors and bias. From 1994 through 1997 the ACAS software package was developed at the USDA Forest Service to implement the Composite Estimator. This paper introduces the Composite Estimator, its application, some of the challenges surrounding its execution, including a brief discussion of the ACAS software development.

Contents

Chapter 1

Motivation for the Composite Estimator in Forest Mensuration

It is important to be able to evaluate and compare the accuracy of different surveying methods used in the estimation of forest resources, especially as cheaper, more advanced techniques become available. The trade-off between accuracy and cost effectiveness must be quantitatively understood in order to make effective forest management decisions.

The Composite Estimator was created to combine separate surveys of a single forest population using different measurement techniques, utilizing the combination of high-accuracy/low-cost and low-accuracy/high-cost methods to form an optimal estimator. The purpose of the composite estimator is to twofold. First it measures the level of agreement between different classification systems which can aid in their evaluation. Second, it makes it possible to combine all available information from different surveys to create a best-possible estimate of the forest composition.

Although the Composite Estimator can be abstracted to almost any combination of measurement techniques and applications in a variety of fields, the remainder of this paper will focus on a single specific implementation involving three set classification systems.

1.1 Terminology

It is handy to formalize some of the vocabulary that will be used throughout this paper. We will be working primarily with normalized vectors whose elements represent proportions of categorical data. The terms “category” and “classification” and “composition” will be used frequently.

Primary Sampling Unit (PSU) A section of land small enough to be classified as falling under one unique category.

Census A compositional measurement of an entire forest population. A census has no associated sampling error.

Category A unique identification or "type" of forest.

Classification The act of measurement by which a forest plot is identified as belonging to a specific category.

Compositional Data Data referring to the composition of a population into its distinct categories.

Composition A normalized vector where each element represents a unique category (or unique combination of categories across different classifications) and measures the proportion of the population that falls under that category.

1.2 Formalizing the Composite Estimator

The ultimate goal is to create estimates of categorical forest composition. In other words, given a specific 100 hectares of a forest, how would that land be broken down into a set number of distinct categories? What percentage of the land could be classified as mature hardwood forest land? How much would be classified as grassland? We want our estimate to be in the form of a normalized vector, where each element represents a distinct forest-type classification.

In this particular application we will define three techniques that are used to gather data:

Satellite Maps Satellite images are obtained of the entire forest population in question. In these images, individual pixels can be mapped to sections of the forest, and the categorical measurement is based on the color of each pixel. This method may have a high error rate but it is unique in allowing a census of the entire forest population to be taken, rather than depending on a sample.

Aerial Photography Aerial photographs are taken of large sections of forest land. Trained specialists carefully examine plots within the photographs and make an ocular estimate of the forest composition. This method is inexpensive enough to allow fairly large samples to be taken.

Ground Survey The most accurate way to classify a forest plot is to send someone in person to visit specific forest locations and personally identify the type of forest. This method is extremely expensive and problematic, because some places are hard to reach. Therefore sampling is often restricted to sampling units that are "within walking distance of an existing road." Due to high costs, these sample sizes are typically small.

With these three methods we will assume that for a specific section of forest the following data have been collected:

Census (Satellite) The forest population in question is mapped to satellite images. Each pixel is matched to a specific type of forest land based on its color. The final statistic is a normalized vector estimating the composition of the entire population. Note that although there may be a substantial measurement (identification) error, there will be no sample error because the entire population is being measured.

Phase I (Satellite and Aerial Sample) A large number of samples are taken from aerial photos. Forest classification is based on visual examination of each PSU in the photographs. Additionally, the sample plot is located on the satellite map and the classification is estimated based on pixel-color. In other words, for each sample in this data set data are gathered (ie. a category is assigned) from both aerial photography and satellite imaging.

Phase II (Satellite, Aerial and Ground Sample) A small number of samples are taken by sending a surveyor to sampled forest locations. (Recall this method is expensive so the sample size will be smaller than that of the Phase I survey.) For every PSU in the Phase II survey all three classification systems are used, meaning that a person is sent out to make a personal inspection, and that plot is also identified on an aerial map and the satellite image. It is important to note that the Phase II PSUs are *not* a subset of the Phase I PSUs. Aerial photograph classification data of the Phase II plots is not included in the Phase I estimation.

1.3 Defining Initial Estimates

Let us say that we wish to estimate the composition of 100 hectares of forest based on four mutually-exclusive categories: hardwood forest, softwood forest, shrub, or other. We want to estimate $\mathbf{x}$ where

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_4 \end{pmatrix}$$

is a normalized vector with four elements representing the proportion of land covered with hardwood forest, softwood forest, shrubs and other. Note these categories are arbitrary, created for this example. The exact designations are irrelevant.

1.3.1 First statistic: Satellite Census

We have three unique data sets available to estimate the forest composition of this population. The first set of data represents the satellite image of this 100

hectare area of forest. This will be in the form of a 4-element vector representing the area that has been identified as hardwood, softwood, shrub and other, with the areas totalling 100 hectares. For example, the data may look like

$$\begin{pmatrix} 23.81 \\ 48.83 \\ 10.12 \\ 17.24 \end{pmatrix}$$

where the units of measure are hectares, and the column sums to 100 hectares. In truth, we are only interested in the proportional composition of the forest, so without loss of generality we can normalize this statistic, dropping the unit of measure (hectares) and thus creating a *compositional* estimate. Henceforth we will assume the estimate $\hat{\mathbf{x}}_{sat}$ is normalized. We will denote the statistic as

$$\hat{\mathbf{x}}_{sat} = \begin{pmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{pmatrix} = \begin{pmatrix} 0.2381 \\ 0.4883 \\ 0.1012 \\ 0.1724 \end{pmatrix}$$

where for each x_i the subscript i is an index that represents the category of hardwood, softwood, etc. If there are m different unique categories (in this example, $m = 4$) then $i \in \{1, 2, ..., m\}$.

1.3.2 Phase I Data: Aerial Photography

The second set of data is a collection of n sample plots. Each plot is measured and categorized by both satellite and aerial photography methods.

We can thus represent the raw data in a table

PSU	Satellite Category	Aerial Category
1	hardwood	hardwood
2	hardwood	softwood
3	softwood	softwood
4	hardwood	hardwood
⋮	⋮	⋮

For each measurement (plot) p where $p = 1, ..., n$ we can represent each PSU by a vector

$$\hat{\mathbf{z}}_p = \begin{pmatrix} z_{11} \\ \vdots \\ z_{ij} \\ \vdots \\ z_{mm} \end{pmatrix} \tag{1.1}$$

where i represents the category (hardwood, softwood, etc.) as identified from the satellite map and j represent the category as identified from the aerial

photography. Note that this vector will have $m-1$ elements equal to zero and a single element equal to one.

The measurements of these n plots can then be used to compute two statistics: a sample mean and sample covariance matrix using straightforward methods. If we combine the vectors $z_1, ..., z_n$ into a data matrix $\mathbf{Z}$ then we can create a sample mean $\hat{\mathbf{x}}_{photo}$ and sample covariance matrix $\hat{\mathbf{S}}_{photo}$ by

$$\hat{\mathbf{x}}_{photo} = \frac{1}{n}\mathbf{Z}'\mathbf{1} \tag{1.2}$$

$$\hat{\mathbf{S}}_{photo} = \frac{1}{n}\mathbf{Z}'\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}'\right)\mathbf{Z} \tag{1.3}$$

These equations are the most basic traditional estimators of a vector mean and covariance that can be found in the first chapter of a multivariate statistics text. In all likelihood the actual sampling design of the Phase I and Phase II surveys will call for more complicated equations than 1.2 and 1.3. For the remainder of this presentation we will assume that the statistician has used the most appropriate estimates $\hat{\mathbf{x}}_{photo}$ and $\hat{\mathbf{S}}_{photo}$.

1.3.3 Phase II Data: Ground Survey

The ground survey data can be represented in a manner almost identical to the aerial photography previously described. For n plots (We're going to reuse the variable n here, where n will be significantly smaller than it was in the Phase I section.) we will have three categories: the satellite classification, an aerial photography classification *and* a ground survey classification.

In this data set we can represent each plot sampling unit p by the vector

$$\hat{\mathbf{z}}_p = \begin{pmatrix} z_{111} \\ \vdots \\ z_{ijk} \\ \vdots \\ z_{mmm} \end{pmatrix} \tag{1.4}$$

where i, j and k represent the categories as identified by the satellite, aerial photography and ground surveys respectively. This is identical to equation 1.1 except that there is one more index. It is therefore possible to follow the same process to arrive at the statistics $\hat{\mathbf{x}}_{ground}$ and $\hat{\mathbf{S}}_{ground}$ for sample mean and covariance matrix for the third data set.

There is one important thing to point out here. The composition estimate $\hat{\mathbf{x}}_{ground}$ is a vector of size m^3. In practice, many of these elements will be equal to zero, because it is unlikely for *every* m^3 combination of categories to occur in the Phase II survey. Ideally we hope that most of the observations will occur where the satellite, aerial photography and ground classifications agree on the same category, so the various $m^3 - m$ *"misclassifications"* (situations where the

three measurement methods don't agree) should ideally be less likely. Note also that the Phase II survey is the one with the smallest sample size due to increased cost.

In practice we will have many zero elements in the composition vectors and zero-valued rows and columns in their corresponding covariance matrices. To simplify computation we drop those zero elements, reducing the sizes of the vectors and matrices. This does not effect the resulting computations.

1.3.4 Understanding the Original Probability Model

We have now defined the population mean $\hat{\mathbf{x}}_{sat}$ from the satellite census and the sample means $\hat{\mathbf{x}}_{photo}$ and $\hat{\mathbf{x}}_{ground}$, but we haven't yet discussed what these statistics are trying to measure. In essence, what is the "truth" we are trying to measure?

To be precise, we are measuring the proportions of the distinct categories *as they would be measured by each of the three systems if each system could measure the entire population.* By this definition $\hat{\mathbf{x}}_{sat}$ has no error because it does measure the exact proportion of categories as they are identified by the satellite system. By establishing this model–estimating the forest types as they would be identified by each system rather than trying to determine the "true" categories of the forest–we are not confounding different classes of errors in our estimates.

Naturally one would be often (but not always) interested in what the "true" forest composition is. Since the ground classification system has the least likely amount of error, one might consider it the closest estimate of the true composition of the forest population.

1.4 Spaces and Translation Matrices

The Census, Phase I and Phase II data all exist in different dimensions. It is very simple to map a higher-dimensional space into a lower-dimensional one with a straightforward linear transformation. For clarity we will first define the three vector spaces we'll be working in, and then we will define three transformation matrices that will be used to map between these spaces.

1.4.1 Vector Spaces

When we use the term "compositions" we are referring to normalized vectors that represent the ratios of categorical data. Compositions can be measured in terms of either a single classification system or (in this example) up to three classification systems. We will define three vector spaces $\wp^1, \wp^2$ and $\wp^3$. The superscript represents the number of classifications being used. The satellite census estimate $\hat{\mathbf{x}}_{sat} \in \wp^1$. The space represents all $m \times 1$ vectors $\mathbf{x}$ where $\mathbf{1}'\mathbf{x} = 1$ and $x_i \in [0, 1]$.

$\wp^2$ is the space that contains all compositions with both satellite and aerial photography classifications. It is therefore a $m^2 \times 1$ dimensional space, similarly having its elements sum to one and all elements $x_{ij} \in [0, 1]$. Similarly, $\wp^3$ is the $m^3 \times 1$ dimensional space containing all possible compositions classified over satellite, aerial photograph, and ground survey techniques.

1.4.2 Translation Matrices

In section 1.3 we have defined three different statistics, two with corresponding covariance matrices, but each having a different number of dimensions. It is possible to create linear transformations on $\hat{\mathbf{x}}_{photo}$ so that it is mapped to the same space as $\hat{\mathbf{x}}_{sat}$, and $\hat{\mathbf{x}}_{ground}$ can be similarly reduced to the same space as either $\hat{\mathbf{x}}_{sat}$ or $\hat{\mathbf{x}}_{photo}$.

We will define linear transformation matrices $\mathbf{H}_1$, $\mathbf{H}_2$ and $\mathbf{H}_3$ where $\mathbf{H}_1\hat{\mathbf{x}}_{photo}$ resides in the same space as $\hat{\mathbf{x}}_{sat}$, $\mathbf{H}_2\hat{\mathbf{x}}_{ground}$ also resides in the same space as $\hat{\mathbf{x}}_{sat}$, and finally $\mathbf{H}_3\hat{\mathbf{x}}_{ground}$ resides in the same space as $\hat{\mathbf{x}}_{photo}$

In summary:

$$\mathbf{H}_1 : \wp^2 \rightarrow \wp^1 \quad \mathbf{H}_2 : \wp^3 \rightarrow \wp^1 \quad \mathbf{H}_3 : \wp^3 \rightarrow \wp^2$$

An Example

To make an extremely simple example, we will define two arbitrary forest types A and B. Let us say that we have a satellite census $\hat{\mathbf{x}} \in \wp^1$ where

$$\hat{\mathbf{x}} = \begin{matrix} sat \\ A \\ B \end{matrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{1.5}$$

and define a Phase I survey $\bar{\mathbf{y}} \in \wp^2$ where

$$\hat{\mathbf{y}} = \begin{matrix} sat & photo \\ A & A \\ A & B \\ B & A \\ B & B \end{matrix} \begin{pmatrix} y_{11} \\ y_{12} \\ y_{21} \\ y_{22} \end{pmatrix} \tag{1.6}$$

The transformation matrix $\mathbf{H}_1$ would be defined as

$$\mathbf{H}_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{1.7}$$

In this way we can use $\mathbf{H}_1$ to reduce $\hat{\mathbf{y}}$ into a 2×1 where

$$\mathbf{H}_1\hat{\mathbf{y}} = \begin{matrix} sat \\ A \\ B \end{matrix} \begin{pmatrix} y_{11} + y_{12} \\ y_{21} + y_{22} \end{pmatrix} \tag{1.8}$$

Now the estimates $\hat{\mathbf{x}}$ and $\mathbf{H}_1\hat{\mathbf{y}}$ share a common dimension (specifically, $\mathbf{H}_1\hat{\mathbf{y}} \in \wp^1$) where their elements reflect the compositional proportions where the satellite classification is A or B.

1.5 The Composite Estimator

The purpose of the composite estimator is to combine two sample composition estimates of the forest population using their corresponding covariance matrices as a weighting method. This is similar in principle to the approach one would take to combine two simple scalar means. If one had two different sample estimates $\hat{\mu}_1$ and $\hat{\mu}_2$ with corresponding variances $\hat{\sigma}_1^2$ and $\hat{\sigma}_2^2$, a combined estimator might be a weighted average of the two

$$\hat{\mu}_3 = \frac{\hat{\mu}_1\hat{\sigma}_2^2 + \hat{\mu}_2\hat{\sigma}_1^2}{\hat{\sigma}_1^2 + \hat{\sigma}_2^2}$$

Let's define the composite estimator as it would be used to combine $\hat{\mathbf{x}}_{photo}$ and $\hat{\mathbf{x}}_{ground}$ using their sample covariance matrices for weighting. Dr. Czaplewski used the concept of the Kalman Filter as the basis for the composite estimator[10], so the following equations will be presented in that form. Since the Kalman Filter can be restated as an MLE, the equations could also be presented in that form. (This is examined in greater detail in Appendix B.)

The composite estimator is defined by

$$\hat{\mathbf{X}}_c = \hat{\mathbf{X}}_{ground} + \mathbf{K}(\hat{\mathbf{X}}_{photo} - \mathbf{H}_2\hat{\mathbf{X}}_{ground}) \tag{1.9}$$

where

$$\mathbf{K} = \hat{\mathbf{S}}_{ground}\mathbf{H}_2'(\mathbf{H}_2\hat{\mathbf{S}}_{ground}\mathbf{H}_2' + \hat{\mathbf{S}}_{photo})^- \tag{1.10}$$

The corresponding covariance matrix is defined as

$$\hat{\mathbf{S}}_c = (\mathbf{I} - \mathbf{K}\mathbf{H}_2)\hat{\mathbf{S}}_{ground} \tag{1.11}$$

The matrix $\mathbf{K}$ is termed the *gain matrix* in the Kalman Filter.

There is one distinct difference between equation 1.9 and the traditional Kalman Filter. Since all composition estimates necessarily sum to 1 (being measurements of proportions) all the $\hat{\mathbf{X}}$ estimates are *necessarily linearly dependent* making their covariance matrices singular. The use of the generalized inverse in equation 1.10 is intended to circumvent this difficulty.

1.6 Evaluating the Composite Estimator

During the course of this research project much effort has been spent evaluating the efficacy and validity of the composite estimator. There are a number of assumptions to take into consideration, and even when these assumptions apply, it may be difficult to attain a sufficient sample size to create meaningful results.

1.6.1 Equivalence to MLE Problem

Kalman Filter application to this problem (the Composite Estimator) may seem artificial since the Kalman Filter is typically associated with time series problems and the Composite Estimator formulates it as a filtering problem within a

single time frame. Fortunately the Kalman Filter formulation turns out to be equivalent to an approach using the context of Maximum Likelihood Estimation. This restatement and its equivalence are given as an exercise in Appendix B.

There is an interesting observation to be made regarding the MLE approach. This approach outlined in Appendix B does not take into consideration the unit-sum constraint of compositional data. Typically the ML approach involving constrained equations requires more complex approaches like Quadratic Programming. However in practice we don't require this more complex approach because the composite estimator, whether written in Kalman Filter or MLE form, always returns a result that is also a valid composition. This means the elements are all non-negative and sum to one. Empirically the only times the composite estimator produced a result that wasn't a valid composition, we were always able to discover difficulties in the original data where the results would obviously be nonsensical. Section 1.6.3 describes some examples.

1.6.2 Multivariate Normal Distribution and Sample Size

The application of the Kalman Filter assumes the estimates (sample means) have gaussian errors. Essentially, we are assuming that by the Central Limit Theorem we will be able to make a multivariate normal assumption. In practice, the sample size must be rather large in order for the individual elements of the mean vectors to approach a normal distribution. Frequently there will be a single rare instance in which a plot has been assigned extremely disparate categories with the different classification methods. For example, in a rare case a plot may be categorized as hardwood by the satellite map, softwood by the aerial photography, and shrub by a ground survey. The likelihood of these "singleton observations" increases greatly as the total number of possible categories increases.

In the previous example we had four different possible categories (hardwood, softwood, shrub and other) when in fact practitioners have attempted to apply the composite estimator to systems with eight, ten, twenty or more possible categories. Note that for ten categories the ground data, having classifications for all three measurement systems, is represented potentially by a 1000 element vector!

Due to the "proportional" nature of the estimates $\hat{\mathbf{x}}_{sat}$, $\hat{\mathbf{x}}_{photo}$ and $\hat{\mathbf{x}}_{ground}$, each element x_i is bound in the range $[0, 1]$. If the dimension of $\hat{\mathbf{x}}$ is m then many of the elements will exist inside the range $[0, 1/m]$. For a sufficiently large dimension the sample size will have to be substantial for the error distribution to appear even vaguely gaussian.

1.6.3 Appropriate Sample Design

Two assumptions must be made when using the composite estimator to combine two different surveys. First, we must assume that the survey was planned and implemented using a satisfactory sampling method. Whether it is a simple

random sample, stratified random sample, or more complex system, the validity of the composite estimator assumes the existence of an appropriately unbiased estimate. Second, sample populations used for the Phase I and Phase II estimates should be the same. In our example where we have a large aerial sample and a smaller ground survey, any significant differences in the populations from which samples are drawn may invalidate the composite estimator.

As previously mentioned, the Phase II sample that includes all three classification methods is often restricted to sampling units located within walking distance of a road. This restriction occurs because it is impossible for a surveyor to reach parts of a forest beyond the reach of roads. Unfortunately, this restriction will bias sample selection toward flat, non-extreme forest locations where roads tend to be built. There may be entire forest categories like "rocky alpine mountain-tops" or the middle of lakes that will exist in the satellite census but will have zero occurrence in the Phase I or Phase II samples. Application of the composite estimator in these situations sometimes leads to mathematical paradox and nonsensical results.

Therefore it is important that the census and the surveys sample the same population and (if possible) use identical sampling methods.

1.7 Inherent Difficulties with Compositional Data

Using the Central Limit Theorem to make a gaussian assumption is tempting, because it is easy to work with statistics using multivariate normal distributions. The use of simple linear transformations are simple and straightforward. Unfortunately, there are inherent difficulties with the restrictions imposed by compositional data, specifically the unit-sum constraint. Karl Pearson[22] first pointed out difficulties inherent with the interpretation of correlations between ratios whose numerators and denominators contain common parts.

Since then, difficulties encountered while trying to interpret these correlations have been described in papers by Chayes[5][6], Krumbein[17], Mosimann[20], and in books by Chayes[7], Le Maître[18] and Aitchison[2]. This section examines the most significant problems.

1.7.1 Linear Dependence

The simple linear dependence of the compositions requires us to deal with necessarily singular covariance matrices. The most obvious solution is to remove an unnecessary element, defining it as a linear combination of the others, in order to achieve linear independence. An exploration of this approach is discussed in section 1.8.1.

1.7.2 Gaussian Assumption

In practice, the elements of a composition vector are very small, positive numbers. By definition they must have non-negative values. Frequently these es-

timates have sufficiently large variances that the lower bounds of their confidence intervals fall well into the negative range. Interpretation of this result is completely meaningless. A patchwork solution to this problem is to consider the individual elements' errors as beta priors to a binomial distribution. This argument and its implementation (which we used as the default method of constructing confidence intervals in the ACAS output) is discussed in Appendix A.

1.7.3 Negative bias difficulty

For an m-part composition we have the restriction

$$\mathrm{cov}(x_1, x_2) + \mathrm{cov}(x_1, x_3) + \cdots + \mathrm{cov}(x_1, x_m) = -\mathrm{var}(x_1) \tag{1.12}$$

Although the correlations between all parts of a composition would be expected to freely span the range [0,1] this restriction requires that at least one of the correlations assumes a negative value. Such forced negative correlation could cause some misinterpretation of the data.

1.7.4 Subcomposition difficulty

In most multivariate situations, an q-part subset of an m-part composition would be expected to preserve the covariance structure. For example, consider a 4-part composition of categories A, B, C and D. Then consider a smaller subset of just the categories A, B and C.

	A	B	C	D
A	1.000	0.267	-0.410	-0.484
B	0.267	1.000	-0.266	-0.419
C	-0.410	-0.266	1.000	-0.490
D	-0.484	-0.419	-0.490	1.000

	A	B	C
A	1.000	-0.066	-0.701
B	-0.066	1.000	-0.665
C	-0.701	-0.665	1.000

This example comes from [2] where the above data come from a hypothetical data set. The correlation matrix on top is taken by calculating all four categories. If we instead consider the composition of only the first three categories, the resulting correlation matrix is given on the bottom. Notice the correlation between categories A and B is positive (0.267) but when removing the fourth category the same correlation between A and B becomes negative (-0.066)!

1.7.5 Basis difficulty

The above correlation matrices were computed from a multivariate sample of unit-sum vectors. When we consider the original data which might be measured

(before the unit-sum normalization is performed) we might expect the sample correlation matrix of the original data to closely resemble the sample correlation matrix of the normalized data. This is not the case, and the two correlation matrices may be dramatically different. This suggests obvious problems when we try to use the covariance or correlation matrices of compositional data to interpret the relationship between two variables.

1.7.6 Null Correlation difficulty

It is traditional to consider null-correlation a good indicator of independence. However, in the situations we've already pointed out, a sample where the categories are truly independent will necessarily have *negative* correlations. It is possible to calculate the naturally occurring negative correlations which would occur under an independence assumption and use this to compare with experimental results, but this problem is also fraught with difficulty.

1.8 Alternate approaches to the Composite Estimator

A significant effort was made in the attempt to find alternate approaches leading to a superior successor to the composite estimator. Although every endeavor lead eventually to a dead-end, the examination was far from exhaustive, and there is still a good chance that an alternative approach could still be found. There were three primary directions that I pursued.

1.8.1 Dimensional Reduction

The most obvious means of reducing the linear dependence of the mean estimates is to create a linear transformation of the estimate into a smaller dimension. In other words, in a vector $\mathbf{x}$ if the elements $x_1, x_2, ..., x_m$ sum to 1, then one could attempt to work with elements $x_1, x_2, ..., x_{m-1}$ knowing that $x_m = 1 - \sum_{i=1}^{m-1} x_i$.

In problems with only one dimension, the solution is easy enough to perform. However, it becomes nontrivial when dealing with three-dimensional ground surveys with elements x_{ijk} where any one of the indices may be the one category that has been removed.

1.8.2 Compositional Data Analysis

In his book, Aitchison[2] provides a good approach to dealing with compositional data–data in the form of normalized vectors representing the composition of some item. One of the primary examples of compositional data that he gives is the result of a standard blood test where some proportion of the blood consists of plasma, another proportion of red blood cells, etc.

His approach to the dilemma is to transform the data vectors $x_1, x_2, ..., x_m$ by dividing each element x_1 through x_{m-1} by x_m and taking the natural logarithm of the ratio. The resulting data exhibit a much better gaussian distribution and the issues of linear dependence are removed.

Unfortunately, the technique cannot be applied to this forest survey data because of the fact that the sample data vectors are sparse with too many zero elements. In essence, it becomes impossible to avoid taking logarithms of zero. Aitchison's method can only be applied to samples where each category is guaranteed representation by some nonzero value. (In an example with blood composition, you will never have a sample with 0% white blood cells.)

1.8.3 Categorical Data Analysis

In many ways the problem of forest population classification appears to lend itself naturally to a categorical data problem, where the various surveys could be broken down as 2- or 3-dimensional contingency tables. It is possible that the entire problem could be rewritten into a loglinear model. This avenue has not been exhaustively explored, but an initial analysis ran into a number of stumbling blocks. A formulation of the weighted average of two loglinear models wasn't straightforward.

Another problem still exists with all the prominent zero elements. (insert quick 4x4 table with the 4 sample categories. point out that many of the 16 cells will probably be empty, and for larger dimension and more categories there will be far more 0 elements.) In Agresti's [1] book §7.7, he deals with the problematic nature of empty cells and sparseness in contingency tables.

Chapter 2

The ACAS software package

2.1 History of the Early Prototypes

The development of a software implementation for the Composite Estimator spans almost four years, beginning in early 1994. The first prototype was a script written by Ray Czaplewski in the Stata programming language. This script performed the basic operations necessary to implement the Composite Estimator for a single data set, but was not flexible enough for general use.

The next prototype was developed in Gauss, a matrix programming language. The original plan was to distribute the Gauss runtime along with the Composite Estimator program for people to use throughout the USDA Forest Service. The prototype was able to perform the basic operations, but it lacked flexibility. There was a need to be able to customize operations and analyses to fit various problems. Gauss was a procedural language that required major rewriting of code each time the order of operations was changed. Some work went into writing a program that would rewrite sections of Gauss code which in turn would be run by the Gauss runtime engine. This effort became too complex and was eventually abandoned.

The nature of the application required a high level of flexibility. Compositional estimates needed to be manipulated in different orders and with many configurable options. It was quickly realized that an object-oriented approach was required where classes could be developed to conceptualize compositions (including the census as a special case) and transformation matrices and various statistical reports. It was important to be able to define an arbitrary number of objects and customize the ways in which they were manipulated.

The C++ programming language was selected to fit these criteria. David Beach was employed to assist in the implementation of the application's design in C++ . The software application was given the name of ACAS as an abbreviation of the term "accuracy assessment".

2.2 ACAS First Iteration (version 1.0)

After spending time looking at commercial C++ matrix libraries it became obvious that our needs exceeded the available offerings. David Beach began the development of basic matrix libraries that would perform all the necessary operations. Some of these libraries were purely mathematical, performing simple matrix algebra functions. Other libraries performed operations specific to the application, like creating transformation matrices to map different composition estimates of different dimensions. In July of 1995 the first 1.0 version of ACAS was completed. The C++ language proved to be flexible enough to accommodate all the challenges in flexibility and customization that we encountered.

Appendix C has an exhaustive description of the initial object oriented structure. A summary of this initial library follows.

Numerical Libraries The first C++ libraries involved defining structures to manage vectors and arrays. Simple operations like matrix multiplication were straightforward in their implementation. More complex functions like singular value decomposition were written using algorithms from Numerical Recipes in C[25].

Mapping between Surveys One of the most difficult tasks involved tracking the categorical indices of the sample estimates and their subsequent composite estimators. Since the $\hat{\mathbf{x}}$ estimates had many zero elements, only non-zero elements were preserved. Each vector had to maintain its associated categorical labels, making it possible to construct the linear transformation matrices between estimates of different dimensions.

Error checking When running various data sets through ACAS many numerical difficulties occurred. (Closer examination usually identified these problems with insufficient sample sizes and situations in which categories would be represented in the census but none of the phases.) A large number of error checking routines were developed to test results. (For example, when a generalized inverse was computed basic properties like $AA^{-}A = A$ and $A^{-}AA^{-} = A^{-}$ would be confirmed.) These tests made it possible to quickly flag problems so that careful examination of the data could be made.

Additional Statistics Dr. Czaplewski's work on the composite estimator included the calculation of a "Kappa Statistic" which would provide a measure of general agreement between different classification systems and conditional probabilities of composition estimates. Routines were developed to implement the optional output of these statistics.

Confidence Intervals It is difficult for a user to glean any information directly from a covariance matrix, so composition estimates would be accompanied by confidence intervals, based on the diagonal elements of the covariance matrices. Unfortunately, since most elements of the composite estimator

(or the original sample means) are very close to zero, confidence intervals based on a gaussian assumption would frequently fall outside of the [0,1] constraint, creating meaningless results.

We developed in ACAS the option of selecting from a variety for methods for calculating confidence intervals. In addition to the traditional gaussian model, an optional beta-binomial model was created which would produce confidence intervals within the [0,1] confines. This model is further discussed in Appendix A.

2.3 Going beyond 1.0: Scripting with Tcl

Up to version 1.0 it was intended that the user would provide data in computations instructions via a customized scripting language. Routines were developed to parse an input file, interpret its instructions and perform the requested operations, but this aspect of the application was difficult and took a disproportionate amount of effort. Soon after the release of version 1.0 the Tool Command Language (TCL) was adopted for the scripting problem. TCL is a scripting language with a very simple and straightforward syntax. TCL is designed to be *extended* where its customized functions can be written in C++ and linked to the TCL runtime. In essence, it is possible to create TCL commands to perform the basic ACAS functions, link the ACAS library to the TCL runtime and have the user write ACAS scripts in TCL.

Further exploration was made into creating a Graphic User Interface for ACAS using Tcl/Tk. Work went into creating advanced graphical visualization techniques for analyzing the data. Unfortunately, there were significant limitations with the Tk graphical libraries. An add-on library called BLT showed some promise for rendering advanced graphical methods, but the library was buggy, unstable fell into unsupported obscurity.

As both Dave Beach and I left the setting of Colorado State University and the USDA Forest Service, work was done to stabilize the ACAS code, create a portable source code that would run under both Linux and the AIX systems of the Forest Service.

2.4 The Future of ACAS

Over the past few years there have been inquiries into the status and future of ACAS. At the end of 2002, Dave Beach expressed an interest in resurrecting ACAS and rewriting it in Python, using the Numerical Python extensions. His opinion was that a fully-functional rewrite of ACAS could be performed using highly optimized numerical libraries. We isolated ourselves in a cabin at Red Feather Lakes and spent a full weekend on the complete rewrite.

To his credit, Dave was right about the flexibility of Python and its speed and power in development. By the end of the weekend most of the basic functionality that we'd spent over a year developing with ACAS 1.0 was complete. A couple

additional evenings made it possible to finish a command parser, and run tests with sample data sets.

From this point the future of ACAS is uncertain. The framework exists to move forward in its development, pending developmental funding. However, it is my belief that the issues discussed in the next section need to be addressed before any further work is justified.

2.5 Final Evaluation of ACAS and the Composite Estimator

Many difficulties exist with the theory and application of the Composite Estimator. Most of the data that were processed with ACAS revealed critical flaws–inadequate sample sizes, over-parameterization, different sampling methods–which resulted in meaningless results. Nevertheless, it is essential to be able to quantify some knowledge of the extent of data misclassification, especially in the analysis and evaluation of different measurement systems[29].

There are still some unexplored avenues in restating the problem as a categorical analysis problem worth investigation, but there is a good possibility that the existing Composite Estimator is the best solution to this problem. Given the high frequency of problems encountered with data processed by ACAS, it seems prudent to suggest further effort be directed toward developing improved planning and sample design. By achieving a balance between sample size and the number of categories, and with early testing of data to assure all samples represent the same population, the Composite Estimator may provide useful measures in accuracy assessment. The following are examples of further directions for this research.

Sample Size and Categorical Dimension The most obvious and frequent problem with ACAS is the risk of over-parameterization. In our example we created four simple categories for forest land: hardwood, softwood shrub, and other. Frequently researchers wish to include as many as ten or twenty different categories in their analysis. Given their budget and planned sample size, this is simply unrealistic!

Tools and methodologies might be developed to plan acceptable sample sizes, including the analysis of preliminary test samples. If the survey has already been taken, it would be advisable to come up with a known limit to the number of categories allowed. High correlations between some categories might suggest ways in which the dimension could be reduced by collapsing multiple categories into specific fewer categories. Tests should be explored to determine whether over-parameterization problems still exist.

Acceptable Sampling Design It is important that the sample selection between surveys use similar methods. For example, if aerial photography is going to include flight plans that cover lakes, provisions need to be

made so the ground survey may also include lakes. A simple hypothesis test may be used to determine whether the aerial photography and/or satellite mapping from two surveys (collapsing the vectors into a common dimension) results in estimates of the same population. In other words

$$\mathbf{H}_o : \mathbf{x}_{photo} = \mathbf{H}_3 \mathbf{x}_{ground}$$

Non-gaussian Models Theoretical exploration should be made to see if the composite estimator may be re-crafted within the auspice of Categorical Data Analysis. The facts that the original data can be best represented by contingency tables and that the $\hat{\mathbf{x}}$ estimates resemble the parameter of a multinomial distribution suggest that a more logical mathematical model may be appropriate, if only the application can be implemented with a reasonable level of complexity.

After this work is done, the outlook of the composite estimator and its application will probably be much brighter. It is still doubtful that useful results could be obtained by a field-researcher as product end-user. The inclusion of a trained consulting statistician with an understanding of this specific application of sample design would be mandatory, as evidenced by the propensity of ACAS to yield questionable results.

Appendix A

The Beta-Binomial Model for Proportional Confidence Intervals

Introduction

Composition estimates, being normalized vectors with a potentially large size, have many elements that are extremely close to zero. Their corresponding variances are often sufficiently large that confidence intervals based on a gaussian error model fall outside of the [0,1] range. Such confidence intervals are nonsensical and only serve to confuse users of the ACAS application. For this reason, an alternative model was proposed using a beta-binomial distribution.

The argument went as follows: each element in the composition vectors may seen to represent the likelihood that given any random sample taken from the entire population, the probability that the sample would be classified as forest type i is giving by x_i. This suggests that the compositions fall under a multinomial distribution, rather than a multivariate normal. Under this assumption, a more logical confidence interval can be obtained by using a beta distribution for the parameter space.

Section A.1 sketches out my first approach to this subject. Section A.2 proposes another approach which looks directly at the distribution of the parameter p. Section A.5 talks about the method used to find the smallest exact confidence interval, once the parameters a and b have been determined for the beta distribution.

The legitimacy of this application is hard to defend beyond the logic stated above. Its pursuit was motivated by the desire to make confidence intervals that "seemed to make better sense". Given the amount of fudging required to get ACAS to provide meaningful results, this application seemed no better or worse than any of the application design.

A.1 The "Scaled-Binomial" Confidence Interval

The primary method for calculating confidence intervals is based off a continuous analogy of a Binomial distribution. Each marginal estimate of a classification proportion is really created by averaging a large number of such proportion estimates generated on each sampled unit (plot). Hence, we have a proportion estimate p which represents one of the parameters from a binomial distribution. We also have the variance *of the parameter estimate.*

A.1.1 The Continuous Binomial distribution

If we examine the probability mass function of a binomial distribution, we have

$$\mathrm{p}(x; n, p) = \binom{n}{x} p^x (1-p)^{n-x} \quad \text{for} \quad x = 0, 1, 2, \ldots, n \tag{A.1}$$

There are a few problems involved with applying this distribution to our model. First, we do not know the appropriate value for n. Next, we are no longer dealing with discrete parameters, so n is a continuous parameter. Let us construct a new "continuous binomial" distribution by first noticing that the binomial coefficient can be expressed in terms of Gamma functions, which are continuous in nature.

$$\binom{n}{x} = \frac{n!}{x!(n-x)!} = \frac{\Gamma(n+1)}{\Gamma(x+1)\Gamma(n-x+1)} \tag{A.2}$$

Hence, equation A.1 is equivalent to the continuous binomial

$$f(x; n, p) = \frac{\Gamma(n+1)}{\Gamma(x+1)\Gamma(n-x+1)} p^x (1-p)^{n-x} \quad \text{for } 0 \le x \le n \tag{A.3}$$

The Beta conjugate prior of the Continuous Binomial

In order to generate a confidence interval for the parameter p, we must first determine the appropriate shape of the conjugate prior. The likelihood equation for p is equal to the continuous binomial density function, but with p as the independent variable and x as a parameter. If we make a change of variables and introduce two new parameters a and b defined by

$$\begin{aligned} a &= np + 1 \\ b &= n(1-p) + 1 \end{aligned} \tag{A.4}$$

then we get the following function for the distribution of p

$$f(p; a, b) = \frac{\Gamma(a+b-1)}{\Gamma(a)\Gamma(b)} p^{a-1} (1-p)^{b-1} \quad \text{for } 0 \le p \le 1 \tag{A.5}$$

The integral of this function over the range $[0, 1]$ is not equal to one. However, if we recall the fact that $\Gamma(a+b-1) = (a+b)\Gamma(a+b)$ then we can divide equation A.5 by $(a+b)$ to get

$$f(p; a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} p^{a-1} (1-p)^{b-1} \quad \text{for } 0 \le p \le 1 \tag{A.6}$$

which is the probability density function for a Beta distribution with parameters a and b. We still do not have enough information to create a confidence interval for p using this distribution because there is no estimate for the binomial parameter n. We do, however, have the variance of the parameter estimate p.

Finding the parameters a and b for the Beta

Recall that a Beta distribution has variance

$$\sigma^2 = \frac{ab}{(a+b+1)(a+b)^2} \tag{A.7}$$

If we make the change of variable from a and b to n and p using equation A.4 we get.

$$\sigma^2 = \frac{(1+n\ (1-p))\ (1+n\,p)}{(2+n\ (1-p)+n\,p)^2\ (3+n\ (1-p)+n\,p)} \tag{A.8}$$

which simplifies to

$$\sigma^2 = \frac{(1+n-n\,p)\ (1+n\,p)}{(2+n)^2\ (3+n)} \tag{A.9}$$

We know σ^2 and p and would like to solve for n. This cannot be done directly, so it is necessary to use Newton's Method. Of course, Newton's Method requires the derivative of equation A.8 which is given by

$$\begin{aligned}\frac{\mathrm{d}\sigma^2}{\mathrm{d}n} = &-\frac{(1+n\,(1-p))\,(1+n\,p)}{(2+n\,(1-p)+n\,p)^2\,(3+n\,(1-p)+n\,p)^2} - \frac{2\,(1+n\,(1-p))\,(1+n\,p)}{(2+n\,(1-p)+n\,p)^3\,(3+n\,(1-p)+n\,p)} \\ &+ \frac{(1+n\,(1-p))\,p+(1-p)\,(1+n\,p)}{(2+n\,(1-p)+n\,p)^2\,(3+n\,(1-p)+n\,p)}\end{aligned} \tag{A.10}$$

This equation simplifies to

$$\frac{\mathrm{d}\sigma^2}{\mathrm{d}n} = \frac{-2-6\,n-2\,n^2+12\,n\,p+2\,n^2\,p-n^3\,p-12\,n\,p^2-2\,n^2\,p^2+n^3\,p^2}{(2+n)^3\,(3+n)^2} \tag{A.11}$$

Once the parameter n has been solved, it is possible to calculate a and b, giving us the exact distribution of the parameter p. From here it is possible to generate a confidence interval for p.

A.1.2 Problems which occur with $\sigma^2 \geq \frac{1}{12}$

Notice in equation A.4 that the parameters $a \geq 1$ and $b \geq 1$. If either a or b is less than one, the parameter n which represents the corresponding continuous binomial distribution would have to be negative! This restricts the variance σ^2 to values less than $\frac{1}{12}$.

Let's examine this in another context. In a worst case scenario in which there is absolutely no data from which to form an estimate $\hat{p}$ then p could have equally likely values over the interval $(0, 1)$. This would be the case where the Beta distribution we were using above was equivalent to a Uniform distribution.

Such a uniform distribution has a variance of $\frac{1}{12}$. If the estimated variance of p is greater than that, then the prior distribution begins to resemble that of a Bernoulli distribution, where p can only take on extreme values of 0 and 1.

A.2 Maximum Likelihood Beta Estimates

Using the argument from section A.1.1, we can see that the likelihood equation of a binomial model is proportional to a Beta distribution function. The method used in the last section began with the statistics $\hat{p}$ and $\hat{\sigma}_p^2$ and came up with estimates a and b for a (Beta) posterior distribution of the parameter p.

The method is ostensibly contrived because instead of beginning with an explicit Binomial distribution $\mathrm{Bin}(\mathrm{n}, \hat{\mathrm{p}})$ of the data, the parameter n has to be estimated with the constraint that the subsequent beta likelihood has a variance equal to $\hat{\sigma}_p^2$.

A more direct solution to this problem is to find the maximum likelihood equations for parameters a and b and use those to estimate the distribution of the parameter p. This approach is difficult to implement, and it suffers from the problem that it requires the sufficient statistics $\sum \ln \hat{p}_i$ and $\sum \ln(1 - \hat{p}_i)$. The only statistics we have to work with, however, are $\bar{p} = \sum \hat{p}_i$ and $\hat{\sigma}_p^2 = \mathrm{Var}(\hat{p})$. This makes maximum likelihood impractical. However, it can be usefull in evaluating other approaches to this problem.

Johnson & Kotz[16] state the maximum likelihood estimators of a and b as the pair of equations

$$\Psi(\hat{a}) - \Psi(\hat{a} + \hat{b}) = \frac{1}{n} \sum \ln p_i \tag{A.12}$$

$$\Psi(\hat{b}) - \Psi(\hat{a} + \hat{b}) = \frac{1}{n} \sum \ln(1 - p_i) \tag{A.13}$$

where $\Psi(\cdot)$ is the digamma function

$$\Psi(x) = \frac{\Gamma'(x)}{\Gamma(x)} \tag{A.14}$$

The solution of these equations must be done numerically.

A.3 Method of Moments Estimation

Probably the most logical (and certainly the most straight-forward) approach to determining the parameters for the Beta distribution uses the moments–specifically the sample mean and variance–to solve for the parameters a and b. Recall that a random variable $p \sim \mathrm{Beta}(\mathrm{a}, \mathrm{b})$ has mean and variance given by

$$\bar{p} = \frac{a}{a + b} \tag{A.15}$$

$$\sigma_p^2 = \frac{ab}{(a + b + 1)(a + b)^2} \tag{A.16}$$

These equations can be solved to find explicit solutions for a and b in terms of E(p) and Var(p).

$$a = \left[\frac{\bar{p}(1-\bar{p})}{\sigma_p^2} - 1\right]\bar{p} \tag{A.17}$$

$$b = \left[\frac{\bar{p}(1-\bar{p})}{\sigma_p^2} - 1\right](1-\bar{p}) \tag{A.18}$$

A.4 Comparison of Three Methods

To compare the three methods (estimate Binomial n, MLE, and Method of Moments) I generated a set of random variables $(X_1, \ldots, X_{10})$ independent identically distributed Bin(100,0.7). From these data, I calculated $\hat{p}_1, \ldots, \hat{p}_{10}$ in addition to the sample mean and variances $\bar{p} = 0.721$ and $s_p^2 = 0.0013211$. The following table gives the estimates for a and b, including the mean and variance of a Beta random variable with those parameters.

	Scaled Binomial	Maximum Likelihood Estimates	Method of Moments Estimates
a	109.316	110.653	109.062
b	42.9142	42.8638	42.2029
mean	0.7181	0.7208	0.7210
variance	0.0013211	0.0013025	0.0013211

A.5 Exact, Smallest Confidence Interval generation

An iterative method is used to generate the smallest possible confidence intervals for the beta prior distribution. The process is iterative, so its drawback is increased processing time. The algorithm which I've developed combines Newton's Method and Bisection to converge upon the solution as quickly as possible.

The routine `betaCI` endeavors to find a value for y such that the two values x_L and x_H (where $y = \text{beta}(x_L, a, b) = \text{beta}(x_H, a, b)$) create bounds for an exact CI level confidence interval.

Appendix B

MLE Equivalence to the Kalman Filter Solution

This appendix will demonstrate the equivalence between the statement of the Composite Estimator in the Kalman Filter context and its statement as a Maximum Likelihood Estimation problem. In order to simplify this exercise, we will use slightly different notation.

B.1 Maximum Likelihood Estimation Approach

Let $\mathbf{x}$ and $\mathbf{y}$ be multivariate normal random variables of dimension p and q respectively $(p > q)$ where $\mathbf{x} \sim \text{MVN}(\mu, \mathbf{\Sigma}_X)$ and $\mathbf{y} \sim \text{MVN}(\mathbf{H}\mu, \mathbf{\Sigma}_Y)$. $\mathbf{H}$ is a linear transformation matrix. We wish to estimate μ common to these distributions. The distributions of these random variables is given by

$$f(\mathbf{x}; \mu, \mathbf{\Sigma}_X) \propto \frac{1}{(2\pi)^{p/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)'\mathbf{\Sigma}_X^{-1}(\mathbf{x}-\mu)\right\} \tag{B.1}$$

$$g(\mathbf{y}; \mathbf{H}\mu, \mathbf{\Sigma}_Y) \propto \frac{1}{(2\pi)^{q/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)'\mathbf{\Sigma}_Y^{-1}(\mathbf{y}-\mu)\right\} \tag{B.2}$$

The likelihood equation given $\mathbf{x}$ and $\mathbf{y}$ and its partial derivative are then

$$\ell(\mu|\mathbf{x}, \mathbf{y}) = -\frac{1}{2}\left\{(\mathbf{x}-\mu)'\mathbf{\Sigma}_X^{-1}(\mathbf{x}-\mu) + (\mathbf{y}-\mathbf{H}\mu)'\mathbf{\Sigma}_Y^{-1}(\mathbf{y}-\mathbf{H}\mu)\right\} \tag{B.3}$$

$$\frac{\partial}{\partial \mathbf{x}}\ell(\mu) = (\mathbf{\Sigma}_X^{-1}\mathbf{x} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{y}) - (\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H})\mu \tag{B.4}$$

A solution will be found by solving $\frac{\partial}{\partial \mu}\ell(\mu) = 0$ or

$$\hat{\mu}_{MLE} = (\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H})^{-1}(\mathbf{\Sigma}_X^{-1}\mathbf{x} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{y}) \tag{B.5}$$

Our aim is to prove that this form is algebraically equivalent to the Kalman Filter estimate (the form used in our Composite Estimator) which can be written in its expanded from as

$$\hat{\mu}_{CE} = \mathbf{x} + \mathbf{\Sigma}_X\mathbf{H}'(\mathbf{\Sigma}_Y + \mathbf{H}\mathbf{\Sigma}_X\mathbf{H}')^{-1}(\mathbf{y} - \mathbf{H}\mathbf{x}) \tag{B.6}$$

In order to do this, we will expand equations B.5 and B.6 to linear combinations of x and y.

From equation B.5 we can expand to

$$\begin{aligned}\hat{\mu}_{MLE} = &\quad (\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H})^{-1}(\mathbf{\Sigma}_X^{-1}\mathbf{x} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{y}) \\ = &\quad (\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H})^{-1}\mathbf{\Sigma}_X^{-1}\mathbf{x} + \\ &\quad (\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H})^{-1}\mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{y}\end{aligned} \tag{B.7}$$

And we can expand equation B.6 to

$$\begin{aligned}\hat{\mu}_{CE} = &\quad \mathbf{x} + \mathbf{\Sigma}_X\mathbf{H}'(\mathbf{\Sigma}_Y + \mathbf{H}\mathbf{\Sigma}_X\mathbf{H}')^{-1}(\mathbf{y} - \mathbf{H}\mathbf{x}) \\ = &\quad \mathbf{x} + \mathbf{\Sigma}_x\mathbf{H}'(\mathbf{\Sigma}_Y + \mathbf{H}\mathbf{\Sigma}_X\mathbf{H}')^{-1}\mathbf{y} - \mathbf{\Sigma}_X\mathbf{H}'(\mathbf{\Sigma}_Y + \mathbf{H}\mathbf{\Sigma}_X\mathbf{H}')^{-1}\mathbf{H}\mathbf{x} \\ = &\quad (\mathbf{I} - \mathbf{\Sigma}_X\mathbf{H}'(\mathbf{\Sigma}_Y + \mathbf{H}\mathbf{\Sigma}_X\mathbf{H}')^{-1}\mathbf{H})\mathbf{x} + \\ &\quad \mathbf{\Sigma}_X\mathbf{H}'(\mathbf{\Sigma}_Y + \mathbf{H}\mathbf{\Sigma}_X\mathbf{H}')^{-1}\mathbf{y}\end{aligned} \tag{B.8}$$

Notice that these two equations B.7 and B.8 are written as linear combinations of $\mathbf{x}$ and $\mathbf{y}$. If we can prove that the coefficients of $\mathbf{x}$ are equal to each other and the same for the coefficients for $\mathbf{y}$, our job is complete. Therefore we will tackle proving the following two equalities

$$(\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H})^{-1}\mathbf{\Sigma}_X^{-1} = \mathbf{I} - \mathbf{\Sigma}_X\mathbf{H}'(\mathbf{\Sigma}_Y + \mathbf{H}\mathbf{\Sigma}_X\mathbf{H}')^{-1}\mathbf{H} \tag{B.9}$$

$$(\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H})^{-1}\mathbf{H}'\mathbf{\Sigma}_Y^{-1} = \mathbf{\Sigma}_X\mathbf{H}'(\mathbf{\Sigma}_Y + \mathbf{H}\mathbf{\Sigma}_X\mathbf{H}')^{-1} \tag{B.10}$$

We will start with B.10. Setting the coefficients of $\mathbf{y}$ in equations B.7 and B.8 we have

$$\begin{aligned}(\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H})^{-1}\mathbf{H}'\mathbf{\Sigma}_Y^{-1} &= \mathbf{\Sigma}_X\mathbf{H}'(\mathbf{\Sigma}_Y + \mathbf{H}\mathbf{\Sigma}_X\mathbf{H}')^{-1} \\ (\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H})^{-1}\mathbf{H}'\mathbf{\Sigma}_Y^{-1}(\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H}) &= \mathbf{\Sigma}_X\mathbf{H}' \\ \mathbf{H}'\mathbf{\Sigma}_Y^{-1}(\mathbf{\Sigma}_Y + \mathbf{H}\mathbf{\Sigma}_X\mathbf{H}') &= (\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y\mathbf{H})\mathbf{\Sigma}_X\mathbf{H}' \\ \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{\Sigma}_Y + \mathbf{H}'\mathbf{\Sigma}_Y\mathbf{H}\mathbf{\Sigma}_X\mathbf{H}' &= \mathbf{\Sigma}_X^{-1}\mathbf{\Sigma}_X\mathbf{H}' + \mathbf{H}'\mathbf{\Sigma}_Y\mathbf{H}\mathbf{\Sigma}_X\mathbf{H}' \\ \mathbf{H}' + \mathbf{H}'\mathbf{\Sigma}_Y\mathbf{H}\mathbf{\Sigma}_X\mathbf{H}' &= \mathbf{H}' + \mathbf{H}'\mathbf{\Sigma}_Y\mathbf{H}\mathbf{\Sigma}_X\mathbf{H}'\end{aligned} \tag{B.11}$$

We will prove equation B.9 in the same manner. In the following proof we will define the following substitution variable

$$\mathbf{V} = (\mathbf{\Sigma}_Y + \mathbf{H}\mathbf{\Sigma}_X\mathbf{H}')$$

Now we can proceed from equation B.9

$$
\begin{aligned}
(\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H})^{-1}\mathbf{\Sigma}_X^{-1} &= \mathbf{I} - \mathbf{\Sigma}_X\mathbf{H}'\mathbf{V}^{-1}\mathbf{H} \\
(\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H})^{-1} &= \mathbf{\Sigma}_X - \mathbf{\Sigma}_X\mathbf{H}'\mathbf{V}^{-1}\mathbf{H}\mathbf{\Sigma}_X \\
\mathbf{I} = \mathbf{\Sigma}_X(\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H}) &- \mathbf{\Sigma}_X\mathbf{H}'\mathbf{V}^{-1}\mathbf{H}\mathbf{\Sigma}_X(\mathbf{\Sigma}_X^{-1} + \mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H}) \\
\mathbf{I} + \mathbf{\Sigma}_X\mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H} - \mathbf{\Sigma}_X\mathbf{H}'\mathbf{V}^{-1}\mathbf{H} &- \mathbf{\Sigma}_X\mathbf{H}'\mathbf{V}^{-1}\mathbf{H}\mathbf{\Sigma}_X\mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H} = \mathbf{I} \\
\mathbf{\Sigma}_X\mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H} - \mathbf{\Sigma}_X\mathbf{H}'\mathbf{V}^{-1}\mathbf{H} &- \mathbf{\Sigma}_X\mathbf{H}'\mathbf{V}^{-1}\mathbf{H}\mathbf{\Sigma}_X\mathbf{H}'\mathbf{\Sigma}_Y^{-1}\mathbf{H} = 0 \\
\mathbf{H}'\left[\mathbf{\Sigma}_Y^{-1} - \mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{H}\mathbf{\Sigma}_X\mathbf{H}'\mathbf{\Sigma}_Y^{-1}\right]\mathbf{H} &= 0
\end{aligned}
\tag{B.12}
$$

Note at this point that for any $\mathbf{H}$, if $\mathbf{Q} = 0$ then $\mathbf{H}'\mathbf{Q}\mathbf{H} = 0$ so we can conclude this proof by showing that $\mathbf{Q} = 0$ hence

$$
\begin{aligned}
\mathbf{\Sigma}_Y^{-1} - \mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{H}\mathbf{\Sigma}_X\mathbf{H}'\mathbf{\Sigma}_Y^{-1} &= 0 \\
\mathbf{I} - \mathbf{V}^{-1}\mathbf{\Sigma}_Y - \mathbf{V}^{-1}\mathbf{H}\mathbf{\Sigma}_X\mathbf{H}' &= 0 \\
\mathbf{I} - \mathbf{V}^{-1}(\mathbf{\Sigma}_Y + \mathbf{H}\mathbf{\Sigma}_X\mathbf{H}') &= 0 \\
\mathbf{I} - \mathbf{V}^{-1}\mathbf{V} &= 0
\end{aligned}
\tag{B.13}
$$

Appendix C

ACAS 1.0 C++ Objects

This appendix comes from the original ACAS Version 1.0 documentation. Its purpose is to outline the initial object-oriented structure of the ACAS application system. After the 1.0 version these objects were left intact, and development proceeded to the process of developing a scripting language to allow a users to use these methods flexibly.

Traditional programming focuses on the procedures and functions which manipulate data. The object oriented programming paradigm introduces a considerable change. The data becomes the focus of the program, and the allowable operations to the data are attached to the data. In "object oriented" terminology, the types of data are called classes, the allowable operations on those data are known as methods, and particular instances of classes (data) are known as objects.

The rest of this section explains many of the classes used in ACAS as well as the methods associated with those classes. It is important to point out that the end user of the ACAS product *does not need to understand the material listed in this chapter!*

The ACAS system utilizes a number of specialized data classes. Each consists of a numerical matrix and some indexing matrices. An indexing matrix assigns categorical strings to each row of the numerical matrix. The following sections will demonstrate this system.

A generic C++ Class object has been created for both numerical and string matrices. [3] The numerical matrix has been constructed with standard operations defined, like matrix (and scalar) multiplication, singular-value decomposition, generalized (Moore) matrix operations and much more. The string matrices have been constructed with operations which perform row sorting operations. From these basic components, the more complex objects are created.

C.1 The Numeric Matrix Class

Perhaps the most basic building blocks of the C++ class system which has been constructed for ACAS is the numeric matrix class. This class of object dynamically stores a matrix of floating point variables in memory and allows standard operations to be performed on it.

C.1.1 Description of Methods

The numeric matrix class in implemented as class which uses strictly *long double* precision math operations on its elements. This means ACAS will use the most precise possible data type, given the architecture of the machine on which it is running. On most modern IBM PC or compatable machines, the long double math operations are preformed by the internal math coprocessor, which uses 80 bit precision (10 bytes). On some UNIX systems, the precision of the long double type is 128 bits (16 bytes).

The defined methods on the numeric matrix include dimensioning (to declare the size of the matrix), printing, and copying. Arithmetic operations are defined for matrices including matrix addition, subtraction, multiplication, scalar multiplication, j matrix addition, and j matrix subtraction. A handful of other useful methods have been defined for the numeric matrix including singular value decomposition, generalized (or Moore) inverse, zero matrix creation, identity matrix creation, row extraction, column extraction, and horizontal and vertical concatenation. Internally, this class is referred to as an "Nmatrix".

C.1.2 Examples

Following is an exhastive list of the defined Matrix operations with small examples included:

(Assume `one`, `two`, `three`, `A`, `B`, and `C` are numeric matrices, x is a long double, and that `cout` is an ostream.)

Dimensioning Methods

dimension to size	`one.dim(4, 5);`
redimension to size	`one.redim(5, 4);`

Note: This has strange effects and only works when the total number of elements is preserved.

Assignemnt and Arithemetic Methods

equality assignment	`one = two;`
element access (one based)	`one(3, 3) = 4.5;`
j-matrix addition	`one = two + 3;`
	`one = 3 + two;`
	`one += 3;`
matrix addition	`one = two + three;`
	`one += two;`
j-matrix subtraction	`one = two - 3;`
	`one = 3 - two;`
	`one -= 3;`
matrix subtraction	`one = two - three;`
	`one -= two;`
scalar multiplication	`one = two  4;`
	`one = 4  two;`
	`one = 4;`
matrix multiplication	`one = two  three;`
	`one = two;`
scalar "division"	`one = two / 3;`
	`one /= 3;`
	`one = 3 / two;`
matrix "division" (all using ginv)	`one = two / three;`
	`one /= two;`

Note: The command `one = 3 / two` replaces `one(i, j) = 3 / two(i, j)` for all i and j in the matrix.

Concatination, Splitting, and Swapping Methods

horizontal concatenation	`one = two | three;`
	`one |= two;`
vertical concatenation	`one = two & three;`
	`one &= two;`
column extraction	`one = two.cols(keyof(1, 2));` [1]
row extraction	`one = two.rows(keyof(1, 2));`
column swap	`one.colswap(2, 4);`
row swap	`one.rowswap(3, 2);`
transposition (in place)	`one.t();`
(functional construction)	`one = t(two);`

Note: Read on to learn more about "Keys" and the keyof function.

Summation and Normalization Methods

column sum	`x = one.col_sum(3);`
row sum	`x = one.row_sum(5);`
sum of all elements	`x = sum(one);`
column normalization	`x = one.col_norm(3);`
row normalization	`x = one.row_norm(5);`

Note: In normalization methods, the normalization factor is stored in x.

Absolute Value, Trace, Square, and Square Root Methods

elementwise absolute value	`one = abs(two);`
(in place)	`one.abs();`
sum of absolute value	`x = sumabs(one);`
trace of a matrix	`x = trace(one);`
square of a matrix	`one = sqr(two);`
(in place)	`one.sqr();`
transpose-square A^-A	`one = trsqr(two);`
(in place)	`one.trsqr();`
square-transpose AA^-	`one = sqrtr(two);`
(in place)	`one.sqrtr();`
Cholesky square root	`one = sqrt(two);`
(in place)	`one.sqrt();`

SVD and Generalized Inverse Methods

singular value decomposition	`one.svd(A, B, C);`
generalized inverse (in place)	`one.ginv();`
(functional construction)	`one = ginv(two);`

Output Methods

output (with field width)	`one.out(8, cout);`
(single row)	`one.outrow(3, 8, cout);`
(part of row)	`one.outrowpart(3, 2, 5, 8, cout);`
(single row, integer format)	`one.outintrow(3, 8, cout);`
(part of row, integer format)	`one.outintrowpart(3, 2, 5, 8, cout);`

Special Matrix Setting Methods

identity matrix	`one.I(5);`
(using previous dimensions)	`one.I();`
	`one = I(5);`
j matrix	`one.J(5, 5);`
(using previous dimensions)	`one.J();`
	`one = J(5, 5);`
zero matrix	`one.Z(5, 5);`
(using previous dimensions)	`one.Z();`
	`one = Z(5, 5);`

Zero Rounding and Equality Methods

round near zero values	`one.round_near_zeros(1.0e-6);`
equality tests	`if(one == two) {···}`
	`if(one != two) {···}`
(within a tolerance)	`if(tol_equal) {···}`

Kronecker Product Methods

Kronecker (tensor) product	`one = kronecker(two, three);`

Number of Rows and Columns Methods

number of rows	`if(one.nrows() > two.nrows()) {···}`
number of columns	`if(one.ncols() < two.ncols()) {···}`

Matrix Destruction Methods

free memory used by matrix	`one.destroy();`

C.2 The String Class

Although ACAS formerly had a string class which was developed along with the main program, as time went on, it became necessary to program in more and more features. We came to the realization that it is not worth it to reinvent the wheel. ACAS now uses the string class described in the GNU G++ [19] library.

C.3 The String Matrix Class

The string matrix class defines a matrix of strings. This is used in the construction of categorical classifications of sampled data. One of the most outstanding features about a string matrix is it's ability to perform a (fast) alphabetical heapsort. In recent development stages, this has made it easier to alphabetize large lists of data quickly. In addition to this, a method has also been added which sorts a list and reduces it to its unique elements. This method is referred to as *reduction* of a list. Reduction is used in many places in the ACAS main code.

C.3.1 Description of Methods

As with the matrix, the string matrix shares the methods of dimensioning, copying, printing, extracting rows and columns, and concatenation with other string matrices. Internally this class is referred to as an "Smatrix."

C.3.2 Examples

Following is a list of the defined Smat operations with examples included:

(Assume `one`, `two`, and `three` are string matrices, that `a_key` is a preloaded key, and that `cout` is an ostream.)

Dimensioning Methods	
dimension to size	`one.dim(4, 5);`
redimension to size	`one.redim(5, 4);`
Equality and Elemental Access Methods	
equality assignment	`one = two;`
element access (one based)	`one(3, 3) = 4.5;`
Concatination, Splitting, and Swapping Methods	
horizontal concatenation	`one = two | three;`
	`one |= two;`
vertical concatenation	`one = two & three;`
	`one &= two;`
column extraction	`one = two.cols(keyof(1, 2));`
row extraction	`one = two.rows(keyof(1, 2));`
transposition	`one = t(two);`
(in place)	`one.t();`
row swap	`one.rowswap(2, 3);`
column swap	`one.colswap(3, 1);`

Output Methods

terminal output (with field width)	`one.out(8, cout);`
(single row)	`one.outrow(3, 8, cout);`
(part of row)	`one.outrowpart(3, 2, 5, 8, cout);`

Special Matrix Setting Methods

blanking a matrix	`one.blank(4, 5);`
(using previous dimensions)	`one.blank();`
	`one = blank(4, 5);`

Sorting and Reduction Methods

alphabetical heapsort	`one = sort(two, keyof(1, 4));`
(in place)	`one.sort(keyof(1, 4));`
alphabetical reduction	`one = reduce(two, a_key);`
(in place)	`one.reduce(a_key);`

Number of Rows and Columns Methods

number of rows	`if(one.nrows() > two.nrows())` {...}
number of columns	`if(one.ncols() < two.ncols())` {...}

String Matrix Destruction Methods

free memory used by matrix	`one.destroy();`

C.4 The Phase Data Class

This object contains the raw data from the file input. Consider each row to be a record of data. Each record represents a unique identification of inventoried forest. The first two or three (really an arbitrary number) columns represent categorical classifications of the sample, ie. mapped classification, photo-interpretation classification, etc. The next column contains the PSU (primary sampling unit) which represents which sampled unit of forest was measured. These columns together form a single string matrix which uniquely identifies each record of the actual data.

The last column is really an $n \times 1$ element numerical matrix which represents the number of sampling units that fall under the classification specified by the same row of the corresponding string matrix. Together, the string matrix (in this example, the Mapped and Ground classifications and the PSU identifier) concatenated with the numerical matrix (the last column) forms the Phase Data Class.

		PSU	COUNT
Mapped1	Ground1	1	3
Mapped3	Ground3	1	7
Mapped4	Ground2	3	2
Mapped2	Ground2	3	6
Mapped1	Ground1	5	7
Mapped2	Ground3	7	2
Mapped4	Ground1	7	2
Mapped2	Ground3	8	4
Mapped1	Ground1	8	16

C.4.1 Description of Methods

The Phase Class includes methods which allows it to create itself from a buffer (listing of string data), and a number of categorical descriptors (also passed to it from the sending code). With this data, the phase class can determine how many records can be made from the buffer, and

create those records internally. If the Phase Class senses that the buffer information is corrupt (usually from an incorrect number of entries), an error is flagged and execution is aborted.

Another feature of the Phase Class is its ability to sort and reduce itself. Once the Phase Class has loaded its information, it can reduce itself, adding the count values on duplicate entries.

Finally, the Phase Class can tabulate itself, creating a table (see next class) with the unique categorical descriptors on the left hand side of the table, and the unique PSU's above the table. Each count is then placed in its appropriate position in the table. The Table Object is then returned. Internally, the Phase Class is referred to as a "Phase."

C.4.2 examples

Here is an exhaustive list of external methods allowed on the Phase data type:

(Assume `pha` is a phase object, that `tab` is a table object, that `the_data` is a dataread object attached to a file with category `#PHASE_I` and that `cout` is of type ostream.)

Creation from a Datafile	
create a phase object	`pha.create(the_data, ``#PHASE_I'');`
Itemwise Reduction and Combination	
reduce to unique entries	`pha.reduce();`
Output Methods	
output to ostream	`pha.out(cout);`
Tabulation of Data	
make table from phase	`tab = pha.tabulate();`
Phase Destruction	
free memory used by phase	`pha.destroy();`

C.5 PSU Ratio Reduction Matrix

The Phase Data object can be reduced to another important structure. Notice in the next table the rows of this matrix are uniquely specified by combinations of the various Mapped and Ground classifications. This example is arbitrary. There can be more than just two types of classification. For example, there could be three columns representing Mapped, Ground, and Surveyed classifications.

		PSU Identifier				
		1	3	5	7	8
Mapped1	Ground1	0.30	0	1.00	0	0.80
Mapped2	Ground2	0	0.75	0	0	0
Mapped2	Ground3	0	0	0	0.50	0.20
Mapped3	Ground3	0.70	0	0	0	0
Mapped4	Ground1	0	0	0	0.50	0
Mapped4	Ground2	0	0.25	0	0	0

Notice that sum of each column is equal to 1. If we instead sum the *rows* we get a measure of forest population that falls under each possible classification.

C.5.1 Description of Methods

Like many of the other classes in ACAS, the ratio reduction matrix has the a method to dimension itself (for on-the-fly creation), as well as methods to print and copy itself. The most impressive feature of the ratio reduction matrix, however, is the ability to create a ratio vector (see next class) class from the data existing in the table. In this process, the table creates both the ratio vector (from the row normalizations of the counts in the matrix), but also the associated variance-covariance matrix. The created ratio is a separate object, and is returned as a result of this process. Internally, the ratio reduction matrix class is referred to as a "Table."

C.5.2 Examples

Due to a lack of time, no examples can be listed in the manual. Examples for the remaining ACAS classes will be added as time permits, and future versions of the documentation will be released.

C.6 Categorical Ratio Vector

If we take the sums of the rows from a PSU Ratio Reduction Matrix and normalize them (divide each by the number of unique PSU Identifiers) we get a vector of ratios, where each element describes exactly what proportion of the total population falls under each forest classification. The Categorical Ratio Vector consists of a string matrix (used to identify a specific forest classification), a numerical matrix — really a vector — which represents the estimated ratios, and a covariance matrix which estimates the covariances between the individual categories. These three objects are separated by double-lines in the following table.

A significant property of the "ratio object" is that the estimated ratio always has a corresponding covariance matrix. As new ratios are made — either by reducing categories or by composite-estimation — their corresponding covariance matrices are simultaneously calculated.

Mapped1	Ground1	0.42	0.17	-0.06	-0.03	-0.02	-0.04	-0.02
Mapped2	Ground2	0.15	-0.06	0.09	-0.02	-0.02	-0.02	0.03
Mapped2	Ground3	0.14	-0.03	-0.02	0.04	-0.02	0.04	-0.01
Mapped3	Ground3	0.14	-0.02	-0.02	-0.02	-0.08	-0.01	-0.01
Mapped4	Ground1	0.10	-0.04	-0.02	0.04	-0.01	0.04	-0.01
Mapped4	Ground2	0.05	-0.02	0.03	-0.01	-0.01	-0.01	-0.01

C.6.1 Description of Methods

The categorical ratio vector has very few sophisticated internal methods. This is because the ratio vector is created by a Table object (the ratio vector lacks a sophisticated method to create itself), and because all major operations involving ratio vectors are methods of the transformation matrix (see next class). The ratio vector has only the very basic abilities of ACAS objects. It has methods to dimension itself, print itself (and print diagnostic warnings, reports and statistics), copy itself, and access some of its internal components. Internally this class is referred to as a "Ratio."

Bibliography

[1] Alan Agresti. *Categorical Data Analysis*. John Wiley & Sons, 1990.

[2] J. Aitchison. *The Statistical Analysis of Compositional Data*. Chapman and Hall, 1986.

[3] Guido Buzzi-Ferraris. *Scientific C++ : building numerical libraries the object-oriented way*. Addison-Wesley, 1993.

[4] Debra Cameron and Bill Rosenblatt. *Learning GNU Emacs*. O'Reilly & Associates, Inc., 1991.

[5] F. Chayes. On correlation between variables of constant sum. *Journal Geophysical Research*, 65:4185–4193, 1960.

[6] F. Chayes. Numerical correlation and petrographic variation. *Journal of Mathematical Geology*, 70:440–452, 1962.

[7] F. Chayes. *Ratio Correlation*. University of Chicago Press, 1971.

[8] R. C. Christensen. *Linear models for multivariate, taime series, and spatial data.* Springer-Verlag, 1991.

[9] J. Cohen. Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70:213–220, 1968.

[10] Raymond L. Czaplewski. Accuracy assesssment of remotely sensed classifications with multi-phase sampling and the multivariate composite estimator. In *16th International Biometric Conference*, volume 2, page 22, Hamilton, New Zealand, 1992.

[11] Raymond L. Czaplewski. Assessment of classification accuracy and extent estimates for a land cover map with double sampling. (submitted to Forest Science), 1994.

[12] Raymond L. Czaplewski. Variance approximations for assessments of classification accuracy. Research Paper RM-316, USDA Forest Service, 1994.

[13] Frank L. Friedman and Elliot B. Koffman. *Problem Solving, Abstraction, and Design Using C++* . Addison-Wesley, 1994.

[14] Jane Hahn. *LaTeX for Everyone*. Personel TeX, Inc., 1991.

[15] Cameron Hughes, Thomas Hamilton, and Tracey Hughes. *Object-Oriented I/O Using C++ IOstreams*. John Wiley & Sons, Inc., 1995.

[16] Norman L. Johnson and Samuel Kotz. *Continuous Univariate Distributions – 2*. Houghton Mifflin Co., 1970.

[17] W.C. Krumbein. Open and closed number systems stratigraphic mapping. *Bull. Amer. Assoc. Petrol. Geologists*, 46:2229–2245, 1962.

[18] R.W. Le Maître. Numerical petrography. Amsterdam: Elsevier, 1982.

[19] Doug Lea. *User's Guide to the GNU C++ Library*. Free Software Foundation, version 2.0 edition, 1992.

[20] J.E. Mosimann. On the compound multinomial distribution, the multivariate β-distribution and correlations among proportions. *Biometrika*, 49:65–82, 1962.

[21] P. L. Patterson and M. S. Williams. Effects of registration errors between remotely sensed and ground data in forest surveys. *Forest Science*, 49(1):110–118, 2003.

[22] K. Pearson. Mathematical contributions to the theory of evolution. on a form of spurious correlation which may arise when indices are used in the measurement of organs. *Proc. R. Soc.*, 60:489–498, 1897.

[23] Greg Perry. *Moving from C to C++ : the ins and outs of object-oriented programming.* Prentice Hall Computer Publishing, 1992.

[24] Richard Petersen. *Introductory C.* Academic Press, Inc., 1992.

[25] William H. Press, Saul A. Teukolsky, William T. Vettering, and Brian P. Flannery. *Numerical recipes in C.* Cambridge University Press, 1988.

[26] Nora Scholin, Martin Sullivan, and Robin Scragg. *OS/2 2.1 Corporate Programmer's Handbook.* IBM, 1993.

[27] Richard M. Stallman. *Using and Porting GNU CC.* Free Software Foundation, version 2.6 edition, September 1994.

[28] Bjarne Stroustrup. *The C++ Programming Language.* Addison-Wesley, 1991.

[29] M. S. Williams and M. Eriksson. Comparison of two paradigms for fixed-area sampling. *Forest Ecology and Management*, 168:135–148, 2002.

[30] M. S. Williams and P. L. Patterson. Comparing area estimation techniques for established surveys using remotely sensed and ground data. *Forest Science: Special Issue on Remote Sensing in Forestry*, 49(3):392–401, 2003.

[31] M. S. Williams, H. T. Schreuder, and R. L. Czaplewski. Accuracy and efficiency of area classifications based on tree tally. *Canadian Journal of Forest Research*, 31:556–560, 2001.