TREE LIST GENERATION DATABASE USER'S GUIDE AND REFERENCE MANUAL

KEVIN R. GEHRINGER

ERIC C. TURNBLOM

STAND MANAGEMENT COOPERATIVE

SMC WORKING PAPER NUMBER 3

AUGUST 2001

COLLEGE OF FOREST RESOURCES

UNIVERSITY OF WASHINGTON

BOX 352100

SEATTLE, WASHINGTON 98115-2100

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The authors wish to thank the Stand Management Cooperative at the University of Washington for providing the funding and data for this project. Specifically, we thank Randol Collier and John Haukaas of the SMC for a wide variety of services throughout the duration of this project. We would also like to thank Dave Marshall, U.S. Forest Service, Pacific Northwest Research Station, Portland, for providing us with much of the initial data that were used to create the tree list generation database, `tgdb1r00`.

We would also like to thank the reviewers of the tree list generation database documentation: Edith C. Sonne, University of Washington, Jim Flewelling, himself, and Temesgen Hailemariam, University of British Columbia. Their comments were most helpful. Any errors remaining in the documentation are attributable only to the authors, after all, we wrote it.

We would also like to thank the Navigation Ancillary Information Facility (NAIF) at NASA's Jet Propulsion Laboratory for their impeccable work in developing and maintaining the SPICE system, an extensive, robust, and portable Fortran subroutine library, which was integral to the development of the tree list generation database software. In particular, we would like to thank Nathaniel Bachman, William Taber, and Charles H. Acton for their support of, and interest in, a nontraditional, i.e., nonaerospace, application of their work.

DISCLAIMER

THERE IS NO WARRANTY FOR THE SOFTWARE OR PROGRAMS PROVIDED, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE OR PROGRAMS (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Chapter 1

INTRODUCTION

Many modern forest stand growth models and forest stand simulation systems use some form of tree list, nominally a set of compatible diameter at breast height (DBH) and height measurements with an indication of tree species, to describe the stand being modeled [11, 20, 8, 35, 36]. An appropriate initial tree list is required as input for individual tree based stand simulation systems in order to accurately predict the stand dynamics and the growth and yield. Tree lists obtained from measurements of existing plots or stands, and for which complete individual tree measurements are available to define the initial condition of a stand, are highly advantageous. With actual tree list data for a stand, forest growth simulations may begin with the *ground truth*. Unfortunately, in many situations the individual tree measurement data which compose a tree list, or from which a tree list may be constructed, are not available or appropriate, e.g., when only stand attributes are available or when actual stand measurement data are not available for the particular management regime under consideration. In order to support the use of tree list based growth and yield or stand simulation models for a broad range of modeling and silvicultural situations, the ability to generate a *reasonable* tree list from a minimal number of stand attributes, e.g., site index, age, number of stems per unit area, etc., is highly desirable.

The use of tree lists as a primary input to forest stand simulators and growth and yield models is likely to persist. Ongoing improvements in growth and yield models and stand simulation systems are increasing the detail or resolution in models

and simulators in order to provide more realistic growth predictions and simulations of stand dynamics. The increases in the detail represented in of the growth and simulation models and in the precision of their outputs are important for economic analyses of forest productivity on a shrinking land base with shorter rotations than in the past, as well as for more ecologically oriented analyses, such as the restoration or creation of habitat structures for wildlife via silvicultural manipulations of forest stands. The models in current use are generally limited by the detail of the initial stand descriptions. Thus, the ability to generate a simulated tree list which mimics the aggregate characteristics of an actual stand is necessary to meet the tree list oriented input requirements of the current and future generations of forest stand growth and yield and simulation models.

A general procedure for creating simulated tree lists will now be briefly described. The procedure is based on combining the representation of a forest stand as a mixture distribution, composed of components representing the tree sizes and the species composition of a stand, and a straightforward database design. The tree list generation procedure and its supporting software and data files will be referred to as a *tree list generation database*. The version of the tree list generation database that is described has been designed for managed, mixed-species stands of Douglas-fir (*Pseudotsuga menziesii*) and western hemlock (*Tsuga heterophylla*) as dominant or codominant species, and possibly with an appropriate mixture of other tree species found throughout Oregon, Washington, and southern British Columbia, west of the Cascade Mountains. Two treatment regimes for managed stands are supported by the tree list generation database: untreated stands and thinned stands.

A conceptual overview of the tree list generation database is provided in the following section, with complete details in Chapter 3. To demonstrate the effectiveness of the proposed tree list generation method, a version of the tree list generation database has been implemented in Fortran 95 and tested with good success. Comprehensive results of the tree list generation database validation tests are presented in Chapter 3.

1.1 A conceptual overview of the tree list generation process

A tree list is a compatible set of diameter at breast height (DBH) and height measurements for each tree in a stand with an indication of tree species. To generate a tree list there are five requirements. First, a method for classifying forest stands so that a simulated tree list may be *customized*, in some sense, to match a specified set of stand level criteria or attributes, e.g., site index, quadratic mean diameter (QMD) and trees per hectare or acre. Second, a method for selecting an appropriate stand class, or set of stand classes, to be used for generating a simulated stand. Third, an appropriate method for representing a forest stand which may be mixed species, multi-aged, etc., in order to generate a tree list. Fourth, a method for generating a tree list from a stand representation; this requirement is necessarily directly related to the third requirement for an appropriate stand representation. Fifth, a wide variety of stand measurement data sets, e.g., measurements of DBH and height with species, that may be used to specify a tree list generation procedure once the first four requirements are met.

These five requirements for generating simulated stands or tree lists may be thought of as defining a database of forest stands that is indexed by a small set of stand attributes. The formulation of the problem concisely defines a framework for its solution, while simultaneously permitting a great deal of flexibility. Additionally, the tree list generation problem and the problem of pseudorandom number generation are quite similar, so the framework of pseudorandom number generation may, and should, be used to guide the development of a tree list generation procedure, particularly since the issues involved in pseudorandom number generation are well understood. In the context of pseudorandom number generation, the tree list generation problem may be thought of as the problem of generating a random stand of trees using a small set of stand attributes as the *seed* of a random stand generation procedure. The combination of these two ideas, a database of stand descriptions indexed

by a set of stand attributes and the framework of pseudorandom number generation, forms the basis for the solution to the tree list generation problem that has been implemented.

There are many possible ways of defining and implementing the first four requirements for generating simulated stands or tree lists, and the subsequent use of the stand measurement data, of requirement five, to calibrate a particular stand generation procedure. The specific methods selected for the first four requirements of generating simulated stands in the tree list generation database are now briefly introduced. For comparison, the equivalent concepts or methods from the classical Weibull parameter recovery method, generally used to represent and simulate diameter distributions [5, 27, 3, 2], are first presented within this context.

1.1.1 Weibull parameter recovery stand generation requirements

The stand classification and stand selection procedures in the Weibull parameter recovery method for diameter distributions are both done implicitly via the multiple regression procedures used to define the Weibull parameter prediction equations. The stand classes and the stands selected for tree list generation are determined by the estimated coefficient values of the Weibull parameter prediction equations. If a broad geographic region is involved in the Weibull parameter recovery method, an explicit classification may first be performed on the stands to reduce the typically large variances that are involved. Weibull parameter recovery procedures are then be applied to each set of explicitly classified stands, producing a set of Weibull parameter prediction equations for each separate class.

The stand representation in the Weibull parameter recovery method for diameter distributions is an explicit, parametric representation that is based on the Weibull probability density function, hence the name. The Weibull probability density function is defined by

$$f(x) = \frac{c}{b}\left(\frac{x-a}{b}\right)^{c-1} \exp\left[-\left(\frac{x-a}{b}\right)^{c}\right], \; a \leq x < \infty$$

where a is the location parameter, b is the scale parameter, and c is the shape parameter [5]. In the Weibull parameter recovery method, the Weibull parameters are estimated for the measured tree diameters to characterize their distribution for each stand measurement data set. The scale and shape parameters are generally the only Weibull parameters estimated, with the location parameter assumed to be zero. Once the Weibull parameters have been estimated for each stand, multiple regression techniques are applied to determine prediction equations for the Weibull parameters based on a small number of stand attributes. The Weibull density function may take on a wide variety of unimodal shapes, which is why it is commonly chosen to represent diameter distributions. Unfortunately, the Weibull distribution is of rather limited utility for representing forestry data. The Weibull distribution is strongly unimodal, and therefore cannot adequately represent multimodal diameter distributions which may occur, nor can it represent relatively flat diameter distributions. In addition, tree species must be accounted for separately, as the Weibull distribution may only be used to represent continuous distributions.

The tree list generation method used in the Weibull parameter recovery method contains three steps. First, the attributes of the desired stand are used with the Weibull parameter prediction equations to obtain a set of Weibull parameters. Second, a Weibull random number generator is used with the parameters obtained in the first step to generate a set of diameter measurements. Three, generate the tree species, if desired, using some appropriate method.

As outlined above, the Weibull parameter recovery approach to generating tree lists has two major drawbacks. First, if new data are to be added to update an existing set of Weibull parameter prediction equations, the entire multiple regression procedure must be repeated with both the original and new data. Second, there is

no clear way to extend the Weibull parameter recovery approach to directly include tree heights and tree species. The tree list generation database has been developed to overcome these difficulties, among others.

1.1.2 Tree list generation database stand generation requirements

The stand classification procedure used for the tree list generation database is an explicit classification based on a multidimensional histogram. The dimensions of the histogram are a set of stand attributes, e.g., site index, QMD, stand type, etc., and the histogram bins define the stand classes. Stands with similar attributes are identified as belonging to the same multidimensional histogram bin or stand class. The multidimensional histogram was chosen as the stand classification scheme primarily for its statistical properties, its ease of implementation, and its overall simplicity and flexibility. In particular, the histogram is a consistent estimator of a probability density function for any data set and for arbitrary dimension; the histogram allows a refinement of its classification if there are sufficient data to warrant the higher resolution; the histogram does not require the reevaluation or reestimation of the classification rules when new data are added.

The stand selection procedure used by the tree list generation database is an explicit procedure which determines the stands most similar to a desired stand based on a specified set of stand attributes, i.e., the nearest neighbor stands as indicated by the set of stand attributes. A similarity function is used to compute similarity scores between a specified set of desired stand attributes and the collection of stand attributes defined by a tree list generation databse stand classification. The similarity function has a value of zero if two sets of stand attributes are identical and increases in value as the sets of stand attributes become more dissimilar. Stands similar to the specified stand will have small similarity scores, scores near zero. The k smallest similarity scores are used to select the collection of stands, or stand classes defined by the multidimensional histogram bins. The k stands or stand classes selected from a

tree list generation database will then be used to form a canonical stand from which a simulated tree list may be generated using the stand representations.

The stand representation chosen for the tree list generation database is an implicit, nonparametric representation based on mixture distributions or densities. A forest stand may be composed of many tree species, with one or more cohorts, or age classes, within each species, and a variety of tree sizes within each species or cohort grouping. This variety in forest stands is nicely captured by the notion of a *finite mixture density* [33, 2, 26, 3]. A finite mixture density is a probability density function of the form

$$p(x \mid (\alpha, \phi)) = \sum_{i=1}^{m} \alpha_i p_i(x, \phi_i))$$

where each $\alpha_i \geq 0, i = 1, 2, \ldots,\ m$, $\sum_{i=1}^{m} \alpha_i = 1$, and where each function p_i is itself a probability density function parameterized by a scalar or vector ϕ_i of parameters [33, 26, 2]. The function p would represent the structure of a forest stand, with each function p_i representing, possibly, a different species, cohort, or size class and the α_i representing the proportion of the stand associated with that particular species, cohort, or size class. The individual probability density functions p_i may be the same e.g., the Weibull density function [2, 3], or the normal density function, but this is not necessary. Each p_i could be specifically selected to suit a particular stand attribute. Additionally, the p_i need not be parametric density functions, like those mentioned, they may be nonparametric density function estimates [28, 9, 10, 26, 24, 25] created for each stand component.

Given this generic form for representing a forest stand, the mixture density may be represented explicitly or implicitly. If the representation is explicit, parametric or nonparametric probability density functions must be chosen for all of the component densities. This involves determining the necessary mixing parameters, α_i, and determining values for the parameters of each parametric density function or nonparametric density function used in the mixture density. The general problem of estimating the parameters of a finite mixture distribution is well understood [26], so

an explicit representation of a stand as a mixture distribution is possible, though it is not necessarily desirable. For example, the explicit mixture distribution approach would severely limit the ability to add new data to a tree list generation database: all of the parameters in the explicit representation would need to be reestimated each time new data were added. Given this difficulty with the explicit representation, among others [28, 33], an implicit representation of the mixture density was chosen for the tree list generation database.

The implicit representation of a forest stand used by the tree list generation database is a finite mixture distribution determined by the actual stand measurement data, DBH and height measurements with an species for each tree. This type of implicit representation of a forest stand is a nonparametric mixture distribution which allows the stand measurement data to *speak for itself*; the actual stand measurement data provide the best, though not necessarily the most concise, representation of a forest stand. Using the actual stand measurement data to represent a forest stand automatically provides the potential for the generation of univariate stands, e.g., tree diameters only, or multivariate stands, e.g., diameter, height, and species, as well as capturing any multimodal characteristics of the forest stands. This approach also provides the potential to generate truly commensurate diameters and heights, if both are desired, with an appropriate species distribution, again if desired, when an appropriate stand generation procedure is used with the data.

The tree list generation method used in the tree list generation database is based upon a method for generating random vectors [31] to augment or appropriately fill in sparse data sets. The random vector generation approach works directly from actual data to generate simulated data points having the same, or nearly the same, covariance structure as the original data. This method is incredibly well suited to the task of generating simulated forest stands, the only requirement being an appropriate list of representative tree measurements for a particular set of stand attributes. The representative stand measurement data may be obtained, as described above, through

a combination of a stand classification procedure (requirement 1) and a stand selection procedure (requirement 2). This method automatically guarantees that individual tree diameters and heights will be compatible, and that the species composition for a stand will be reasonable; they are derived from *actual* data.

In addition to the five requirements for the generation of simulated stands or tree lists, there are two fundamental assumptions and a precise definition of a forest stand that are necessary and will be used throughout this document.

Forest Stand A spatially contiguous group of trees and associated vegetation having similar structures and growing under similar soil and climate conditions [22], or a relatively homogeneous forested area.

Fundamental Assumption 1 (**FA1**) Forest stands, or sample plots, having similar attributes, e.g., site index, QMD, species composition, etc., are similar.

Fundamental Assumption 2 (**FA2**) Attributes of forest inventory data collected from a representative set of sample plots within a larger stand can be extended to the whole stand.

The definition of a forest stand and both of the fundamental assumptions are in widespread use within the science of forestry. The first assumption is a basic tenet of forestry modeling, including growth and yield modeling, though it is frequently not stated explicitly. The second assumption is commonly used in forest inventory development; data from cruised sample plots are scaled up to represent the actual inventory on larger homogeneous forested areas. The second assumption is also generally not stated explicitly, but is presumed by the statistical methods used for the inventory sampling and analysis. These two fundamental assumptions and the definition of a forest stand are presented for clarity and completeness, since they provide the motivation for the tree list generation procedures used in the tree list generation database.

In developing the tree list generation procedures, the first fundamental assumption, **FA1**, is used in its traditional sense, but the second fundamental assumption, **FA2**, is not. **FA2** is traditionally used to scale aggregate sample plot data up to a larger area for inventory purposes. This assumption, however, must also be valid when considering the individual trees on the inventory sample plots, and it is in this sense that the assumption is used for the tree list generation database. Under this interpretation, the plot data are scaled to a larger area by generating sufficient trees to be representative of the larger area, that is, by populating the larger area with individual trees. This interpretation of **FA2** provides the basis for the tree list generation procedures developed for the tree list generation database.

The concept of a *forest stand*, or *stand* of trees, is used in four distinct, but compatible, ways within this documentation. First, *forest stand* may refer to an actual stand of trees, as in the definition of a forest stand just given. Second, *forest stand* may refer to the trees on a temporary or permanent sample plot that is assumed to be representative of a larger area. Third, *forest stand* may refer to a tree list consisting of DBH and height measurement with species, whether actually measured on a sample plot or simulated, that is assumed to be representative of some actual stand of trees. Fourth, *forest stand* may refer to a canonical stand, or multidimensional histogram bin, from the stand classification in a tree list generation database. The usage in each specific situation is consistent with the typical meaning and usage, and so should cause no confusion. Reference to the specific context in which the concept of a *forest stand* is used will be made if the particular cantext is important or to clarify any potential ambiguity.

1.2 Conventions and notations

The following typographic conventions will be followed throughout the remainder of this document.

1. Data file contents and source code when presented in the text will appear in a teletype or fixed pitch font, e.g., `SUBROUTINE XYZ(A,B,C)`.

2. Species common names, e.g., Douglas-fir, will appear in the standard font, and species latin names, e.g., *Pseudotsuga menziesii* will appear in an italic font.

3. References to file names or program names will be in a teletype or fixed pitch font, e.g., `input.dat` or `TGNEW`.

4. References to interest addresses , email addresses, and uniform resource locators (URLs) will be in a teletype or fixed pitch font `http://www.website.com` or `emailhere.there.com`.

5. In examples, input that a user would type on a command line will be in a teletype or fixed pitch font preceded by a prompt indicator, e.g., `PROMPT> dir *.dat`. Output from commands that would be displayed on the computer screen or shell will appear in an italic teletype or fixed pitch font, e.g., *`File successfully created`*.

6. A forward slash , '/', is used as the path element separator throughout this document for consistency. When using the tree list generation database software, either a forward slash, '/', or a backslash, '\', may be used in filenames or directory paths to separate the individual path elements. The software will substitute the correct path separation character for the computing platform being used.

1.3 System requirements

Developed for windows 95/98/NT or later systems using pentium processors. The executables may work on i486DX based systems, but these are considered to be an

obsolete system configuration. The programs have been compiled with pentium optimizations, so performance on i486DX based systems, if the programs execute, may be degraded. The tree list generation database software uses dynamic memory allocation, so there is no upper limit to the data base size, in theory, that may be created and used. The recommended amount of memory is 32 megabytes, but more is always better.

1.3.1 System requirements for standard distribution

1. Approximately 15 MB in a `.zip` file.

2. Approximately 30 MB uncompressed.

1.3.2 System requirements for developer distribution

1. Approximately 45 MB in a `.zip` file.

2. Approximately 90 MB uncompressed.

1.3.3 Contents of standard distribution

1. Tree list generation database executables.

2. The initial tree list generation database which supports untreated and thinned stands of Douglas-fir *Pseudotsuga menziesii*, western hemlock *Tsuga heterophylla*, or combinations of these two species.

3. Tree list generation database documentation.

1.3.4 Contents of developer distribution

1. Tree list generation database executables.

2. The initial tree list generation database which supports untreated and thinned stands of Douglas-fir *Pseudotsuga menziesii*, western hemlock *Tsuga heterophylla*, or combinations of these two species.

3. Tree list generation database documentation.

4. Tree list generation database source code and libraries.

To make full use of the tree list generation database developer distribution a Fortran 90/95 compiler is required. The Lahey/Fujitsu Fortran 95 compiler, LF95 version 5.5, was used as the development compiler [30]. If a different Fortran 90/95 compiler is used some code modifications will likely be necessary to build the tree list generation database libraries and executable programs. In addition, a `make` utility and other `UNIX` based text processing tools would likely prove useful.

1.4 Limitations of the tree list generation database software

1. Douglas-fir and western hemlock are currently the only tree species that may be used to determine stand type, e.g., pure (species basal area $\geq 80\%$) or dominant (species basal area $\geq 50\%$). It is a straightforward, though not trivial, extension of the tree list generation database subroutine library to support arbitrary dominant and codominant tree species for the stand type determination.

2. File format conversion utilities to convert the tree list files produced by the tree list generation database software into formats appropriate for direct input to growth and yield models are not provided. Fortran 90/95 subroutines which read and write each file format are provided to facilitate the development of conversion utilities by those best able to develop them, e.g., the organizations developing growth and yield models or other forestry related software.

3. The methodology developed to generate a tree list currently requires that a user of the system completely specify a standard stand description via a set of input index parameters. This is, in general, not a major constraint because the overall methodology is sufficiently general that an additional software module could be developed and added to an appropriate place in the tree list generation process to estimate any possibly missing parameters.

4. The tree list generation database software performs Imperial to metric or metric to Imperial units conversions. All of the conversions used are accurate to at least 10^{-6} units. This issue is important because all of the data stored in a tree list generation databse are in metric units. For the typical use the accuracy of the conversions poses no concerns, but it is possible to generate tree lists for very large areas, and inaccuracis could appear in this setting.

5. There were several tree species code conflicts in the data, i.e., two different codes being used to represent the same tree species. To resolve these conflicts, the code used by the SMC was selected for use in the tree list generation database. See Section 2.11.4 and Table 2.17 for more information.

    - The tree species code `OH` for other hardwood was changed to be `HD`, the code used by the SMC. Further use of the tree species code `OH` is not recommended.
    - The tree species code `PM` for pacific madrone (*Arbutus menziesii*) was changed to be `MD`, the code used by the SMC. Further use of the tree species code `PM` is not recommended.
    - The tree species code `O3` appeared in some of the data. A description of this species code could not be found, so it was changed to `OT`, the code used by the SMC for unrecognized species or mixtures of species. Further use of the tree species code `O3` is not recommended.

6. Program command line and input/output limitations.

   - Maximum command line length: 4096 characters.
   - Maximum filename length: 1024 characters.
   - Maximum line length for reading or writing files: 16384 characters.
   - Tree list generation database size is limited only by available disk space.
   - The size of a simulated tree list is limited only by the available memory and disk space.

1.5 Installation instructions

In the following sections basic installation instructions are provided for the standard and developer tree list generation database distributions. The installation instructions are provided as a basic list of steps that should be followed to install the software and database. Installation details for any particular operating system are not discussed; thus a basic familiarity and facility with the operating system used is presumed for the installation procedures. If the instructions are followed, no warning or error messages appear. If such a message does appear, please refer back to the installation instructions for clarification.

1.5.1 Installation of the standard distribution

1. Decide where the tree list generation database executable files, documentation, and data files should reside on the hard drive, creating any directories if necessary.

2. Create a temporary directory, `tlgtemp`,

   ```
   PROMPT> mkdir tlgtemp
   ```

where the tree list generation database `.zip` file archive will be placed and its files extracted. This is done to make cleanup easier in the event of an error of some sort.

3. Download the program `unzip.exe` from the download page of the tree list generation database web site, placing it into the the temporary directory `tlgtemp`.

4. Download the standard distribution of the tree list generation database from download page of the tree list generation database web site, the filename is `tlgv1r0s.zip`, also placing it into the the temporary directory `tlgtemp`. A logon ID and password are required for this download operation.

5. Extract the files and directories for the standard distribution `tlgv1r0s.zip` using the program `unzip.exe`,

   ```
   PROMPT> unzip tlgv1r0s
   ```

   which will automatically preserve the directory structure of the distribution. This creates a directory tree in the current directory with the top level directory name `tlgv1r0s`, which contains subdirectories for the documentation, the executable files, and a tree list generation database. The directory `tlgv1r0s` should contain the following subdirectories.

   - `data` This directory contains the various data files used to create the tree list generation database `tgdb1r00` and the tree list generation database itself.
   - `doc` This directory contains the documentation for the tree list generation database software and files.
   - `exe` This directory contains the executable programs which comprise the interface to a tree list generation database.

6. Place the executable files from the directory `tlgv1r0s/exe` into a directory which is on the executable file search path or add the path `tlgv1r0s/exe` to the executable file search path.

7. Place the documentation files from the directory `tlgv1r0s/doc` into the directory for the tree list generation database documentation.

8. Place the tree list generation database directory, `tlgv1r0s/data/tgdb1r00`, and all of its subdirectories, into the location on the hard drive where the tree list generation database should reside, creating a directory if necessary.

1.5.2 Installation of the developer distribution

1. Decide where the tree list generation database executable files, documentation, source code, and data files should reside on the hard drive, creating any directories if necessary.

2. Create a temporary directory, `tlgtemp`,

   ```
   PROMPT> mkdir tlgtemp
   ```

   where the tree list generation database `.zip` file archive will be placed and its files extracted. This is done to make cleanup easier in the event of an error of some sort.

3. Download the program `unzip.exe` from the download page of the tree list generation database web site, placing it into the the temporary directory `tlgtemp`.

4. Download the developer distribution of the tree list generation database from download page of the tree list generation database web site, the filename is `tlgv1r0d.zip`, also placing it into the the temporary directory `tlgtemp`. A logon ID and password are required for this download operation.

5. Extract the files and directories for the developer distribution `tlgv1r0d.zip` using the program `unzip.exe`,

   ```
   PROMPT> unzip tlgv1r0d
   ```

   which will automatically preserve the directory structure of the distribution. This creates a directory tree in the current directory with the top level directory name `tlgv1r0d`, which contains subdirectories for the documentation, the executable files, the source code and libraries, and a basic tree list generation database. The directory `tlgv1r0d` should contain the following subdirectories.

   - `data` This directory contains the various data files used to create a tree list generation database `tgdb1r00` and the tree list generation database itself.
   - `doc` This directory contains the documentation for the tree list generation database software and files.
   - `exe` This directory contains the executable programs which comprise the interface to a tree list generation database.
   - `libs` This directory contains the object libraries used to build the executable programs.
   - `src` This directory contains the Fortran 95 source code for the tree list generation database executables, library, and supporting subroutines.

6. Place the executable files from the directory `tlgv1r0d/exe` into a directory which is on the executable file search path or add the path `tlgv1r0d/exe` to the executable file search path.

7. Place the documentation files from the directory `tlgv1r0d/doc` into the directory for the tree list generation database documentation.

8. Place the tree list generation database source code directory, `tlgv1r0d/src`, and all of its subdirectories, into the location on the hard drive where the tree list generation database source code should reside, creating a directory if necessary.

9. Place the tree list generation database directory, `tlgv1r0d/data/tgdb1r00`, and all of its subdirectories, into the location on the hard drive where the tree list generation database should reside, creating a directory if necessary.

1.6 Software bug reporting and contact addresses

In the event that a software bug is discovered, send an email or FAX to the Stand Management Cooperative at the University of Washington. Before sending the problem report to the SMC, please use the following short check-list when formulating your problem report. This will enable a more rapid resolution of the problem.

1. Be sure that the problem to be reported is real and not simply an erroneous input value or some other simple problem that is not a bug or defect.

2. Attempt to fully, and concisely, document the circumstances under which the problem occurred. Be sure to include the following items to include in any message reporting a possible bug that is sent to the SMC.

    - Your name and contact information, email address and phone number in particular.
    - The name and version of the program that was used.
    - The version of the tree list generation database subroutine library that was used.
    - All input or output files that were used when the problem occurred.
    - Error messages, if any, reported by the tree list generation database software or the computer operating system when the problem occurred.

3. The subject line of the email or FAX should clearly indicate that the message is a bug report or problem report for the SMC tree list generation database software.

Contact the Stand Management Cooperative at the address below to report problems with, or possible bugs in, the tree list generation database software.

Stand Management Cooperative
University of Washington
College of Forest Resources
Room 164 Bloedel Hall
BOX 352100
Seattle, WA 98195-2100

email: `silvproj@u.washington.edu`
FAX: (206) 685-3091

SMC Home Page:
`http://www.cfr.washington.edu/smc/`

SMC Tree list generation database Home Page:
`http://depts.washington.edu/silvproj/tlghome/`

Chapter 2

TREE LIST GENERATION DATABASE USER'S MANUAL

This chapter describes the structure of a tree list generation database and the executable programs and files necessary to effectively use the tree list generation database system. The chapter begins with a brief introduction to the tree list generation database. Next, the software used to access and manipulate a tree list generation database is described, and examples are given for each executable program. The tree list generation database input and output file descriptions and formats follow the program descriptions. The chapter ends with a brief summary of the data contained in the initial tree list generation database `tgdb1r00`.

2.1 Introduction to the tree list generation database

The tree list generation database consists of a set of eight executable programs and a customized database of tree measurement data, compatible DBH and height measurements with tree species, that may be used to construct simulated tree lists. The tree list generation database permits the construction of simulated forest stands or tree lists using aggregate or per unit area descriptions of forest stands. Stand descriptions are provided, for example, as site index, stand total age, QMD stand type, and trees per hectare (acre). This chapter describes a particular implementation of a tree list generation database, including a basic description of the database structure, the database access and utility programs, and the data files that are used to generate tree lists.

The eight tree list generation database programs are command line oriented, and have their own online help that may be accessed by using the appropriate command line option, `-h` or `-help` with no other command line options. The tree list generation database programs provide the primary means of access to a tree list generation database, and provide a basic set of database oriented functions. The tree list generation database programs and their functions are briefly described below, with complete descriptions in Section 2.2 to Section 2.8. The majority of this chapter is devoted to describing the individual tree list generation database programs, their use, and their input and output files.

`TGNEW` Create a new tree list generation database. Section 2.2.

`TGADD` Add a stand measurement file or a set of stand measurement files to a tree list generation database. Section 2.3.

`TGRAND` Generate a simulated stand or random stand measurement file from a tree list generation database stand description file, or generate a set of simulated stands from a set of stand description files. Section 2.4.

`TGSUMRY` Summarize a tree list generation database. Section 2.5.

`TGLOCK` Lock a tree list generation database to prevent further modifications, e.g., the addition of new stand measurement data. Section 2.6.

`TGUNLOCK` Unlock a tree list generation database to allow further modifications, e.g., the addition of new stand measurement data. Section 2.6.

`TGXPLODE` Break a tree list generation database into its component parts. This is necessary if the index parameters in an existing database are to be changed, or if the stand classification interval widths are to be modified. Section 2.7.

`TGSCATRP` Create a file of scatter plot data for a treatment in a tree list generation database from the multidimensional histogram based stand classification. Section 2.8.

The eight tree list generation database programs each use one or more of the following tree list generation database input or output files. The tree list generation database input and output files have been designed to be both easily scanned by software and easily understood by human beings. The files generally follow a *keyword equals value* format and may be viewed or edited using any text editor.

stand measurement file A tree list generation database stand measurement file defines the attributes of a measured stand of trees, including DBH and height measurements and an indication of species for each tree. Section 2.11.1.

stand description file A tree list generation database stand description file defines the attributes of a measured stand that are desired for a simulated or random stand that is produced by `TGRAND`. Section 2.11.2.

random measurement file A tree list generation database random measurement file is identical in format to the stand measurement file except that it is produced by `TGRAND` to simulate a stand specified in a stand description file. Section 2.11.3.

species mapping file A tree list generation database tree species mapping file defines a mapping from short character tree species identifiers to integer codes used internally by the tree list generation database software. This file is only used when creating a new tree list generation database. Section 2.11.4.

schema file A tree list generation database schema file defines the index parameters that are to be used in a particular tree list generation database. This file is only used when creating a new tree list generation database. Section 2.11.5.

In addition to the tree list generation database files just described, there are two output file types, a summary file and a scatterplot file, and one input file type, a list file, that are used by the tree list generation database software. The tree list generation database summary and scatterplot fils are used to represent summaries of the stand classification structure, the treatment types, and the individual tree data contained in a tree list generation database. The list file is used to provide a list of stand measurement or stand description file names to be processed in a batch to either `TGADD` or `TGRAND`, respectively. The files specified in the list file for each program must be of the appropriate type: stand measurement files for `TGADD` and stand description files for `TGRAND`.

summary file A tree list generation database summary file is a text file that is produced by `TGSUMRY`. The text file contains a concise summary of the data contained in a tree list generation database.

scatter plot file A tree list generation database scatterplot file is a comma delimited text file created by `TGSCATRP`. The scatterplot file allows the individual data dimensions of the multidimensional histogram bin centers from the stand classification to be plotted against each other for each treatment. This file may be thought of as providing a detailed summary of a tree list generation database, and it may be used to identify gaps in the data coverage.

list file A text file that contains a list of file names, one per line, specifying a set of stand measurement files or stand description files that are to be processed by `TGADD` or `TGRAND`, respectively. The file may contain comments, indicated by a percent symbol (%) as the first nonblank character of a line, or blank lines.

2.1.1 Tree list generation database naming conventions

A tree list generation database consists of two components: a software component consisting of source code and executable programs and a data component which stores the tree data used to generate tree lists. In determining suitable naming conventions for a tree list generation database distribution both of these components must be taken into account, as well as the type of tree list generation database distribution: standard or developer. The software version and the database format *must* be compatible, and changes to the contents of the tree measurement data in a tree list generation database should also be taken into account. An appropriate naming convention takes both of these issues into consideration.

A two part naming convention composed of a version number and a revision number has been selected as the naming convention for the tree list generation database software and data components. A third part is added to the name to indicate the type of tree list generation database distribution. Version numbers must agree to guarantee that the software and the data for a particular tree list generation database are compatible. The version number is only changed if there is a major modification to the tree list generation database software, i.e., a modification which affects the file formats or which introduces a potential backward incompatibility. Revision numbers indicate the addition of new stand measurement data to an existing tree list generation database or a backward compatible update, whether maintenance or bug fixes, to the tree list generation database software.

A tree list generation database name in the standard or developer distribution follows the convention `tgdbXrYY`, where `X` is replaced with a single digit version number for the software, and data formats and `YY` is replaced with the revision number, e.g., `tgdb1r00`. After each addition of data to a tree list generation database the revision number should be incremented by one (1) to indicate that the tree measurement data has changed from a previous tree list generation database distribution.

The tree list generation database distribution naming convention is `tlgvXrYD`, where `X` is replaced with a version number for the software and data formats, `Y` is replaced with the revision number, and `D` is replaced with the letter `d` or the letter `s` for the developer or standard tree list generation database distributions, respectively, e.g., `tlgv1r0s`. The revision number in the tree list generation database distribution name should only be incremented if the tree list generation database software changes. The version number should onlybe incremented if there is a major modification to the tree list generation database software.

An existing tree list generation database may be renamed, for example to increase its revision number to reflect the addition of new data. The renaming consistes of two steps. First, change the basename of the tree list generation database information file to the new tree list generation database name. Second, change the tree list generation database directory name to the new tree list generation database name. As an example, suppose new data have just been added to the tree list generation database `tgdb1r00`, and its revision number is to be incremented from `00` to `01` to reflect this modification. Assuming the tree list generation database to be renamed is in the current directory, it is renamed by the following two steps.

1. Change the name of the tree list generation database information file from `tgdb1r00/tgdb1r00.dat` to `tgdb1r00/tgdb1r01.dat`

2. Change the tree list generation database directory name from `tgdb1r00` to `tgdb1r01`.

See Section 2.1.2 for details of the tree list generation database file and directory structure.

2.1.2 Tree list generation database files and directory structure

A tree list generation database consists of a main database directory which contains a species mapping file, a database information file with a basename that is the same as the main database directory name, and a subdirectory for each available treatment. The species mapping file defines which tree species are recognized by a tree list generation database. The database information file contains data relating to the current state of a tree list generation database, e.g., the number and type of treatments, the number of trees in each treatment, the number of histogram bins used to classify stands for each treatment, the index parameters used for the stand classifications, and the creation and last modification dates. Each treatment subdirectory contains four files: a tree data file which contains the DBH, height, and species for all of the trees associated with the treatment; a stand catalog file storing the stand description data for each stand measurement file associated with the treatment, an index file based on the selected index parameters for the treatment, and a file linking the index to the tree data.

The directory structure for the tree list generation database `tgdb1r00` which contains untreated stands and thinned stands as its treatments is given by Figure 2.1. For this tree list generation database, the data for the untreated treatment is stored in the subdirectory `u` and the data for the thinned treatment is stored in the subdirectory `t`. The files in the example which correspond to the description of a tree list generation database given above are given in Table 2.1.

2.1.3 Tree list generation database units conversions

The tree list generation database software provides seamless measurement units conversions between the Imperial and metric systems of measurement for a standard set of forestry measurement data types. The conversion factors used provide a conversion accuracy of at least 1.0×10^{-6} for a value of one (1) unit. The units conversions

```
tgdb1r00/spmap.dat
tgdb1r00/tgdb1r00.dat
tgdb1r00/u/pbktmap.dat
tgdb1r00/u/pindex.dat
tgdb1r00/u/stnddesc.dat
tgdb1r00/u/treedata.dat
tgdb1r00/t/pbktmap.dat
tgdb1r00/t/pindex.dat
tgdb1r00/t/stnddesc.dat
tgdb1r00/t/treedata.dat
```

Figure 2.1: Example of the tree list generation database file and directory structure for the tree list generation database `tgdb1r00` containing both untreated and thinned treatments.

Table 2.1: File descriptions for the tree list generation database file and directory structure for the example in Figure 2.1. The file associations for the thinned treatment type are identical to those of the untreated treatment type, and so are not listed. They are obtained by replacing the `u` with a `t`.

File name	Tree list generation database file description
`tgdb1r00/spmap.dat`	Species mapping file
`tgdb1r00/tgdb1r00.dat`	Database information file
`tgdb1r00/u/pbktmap.dat`	Index to tree data mapping for untreated stands
`tgdb1r00/u/pindex.dat`	Index file
`tgdb1r00/u/stnddesc.dat`	Stand catalog data file for untreated stands
`tgdb1r00/u/treedata.dat`	Individual tree data for untreated stands

performed by the tree list generation database software are listed in Table 2.2.

Table 2.2: The measurement unit equivalents for the Imperial and metric measurement systems for each of the tree or stand measurements recognized by the tree list generation database software.

Measurement Type	Imperial Unit	Metric Unit
diameter	inches (in)	centimeters (cm)
height	feet (ft)	meters (m)
area	acres (ac)	hectares (ha)
basal area	square feet/acre (ft^2ac^{-1})	square meters/hectare (m^2ha^{-1})
density	trees per acre (tpa)	trees per hectare (tph)

2.2 Creating a tree list generation database

The tree list generation database program `TGNEW` is used to create a new tree list generation database. Required inputs are a tree list generation database schema file, which defines the index parameters that are valid for the new database, a tree list generation database species mapping file, which defines the tree species that will be recognized by the tree list generation database programs, and the name of the new tree list generation database. Descriptions of the schema file and the species mapping file may be found in Section 2.11.5 and Section 2.11.4, respectively.

2.2.1 The `TGNEW` program

`TGNEW` is a program that creates a new tree list generation database. Required inputs for this program are the tree list generation database name, a tree list generation database schema file, and the name of a tree list generation database species mapping file. An optional input to the program is the name of a directory where the tree list generation database is to be located. Path elements are limited to eight (8) characters with filenames limited to an eight-dot-three format with eight (8) characters forming the base filename, followed by a dot or period, followed by a three (3) character filename extension. Filenames and path elements may not contain any blanks or other nonprinting characters. These common restrictions are imposed for the easy portability of filenames and directory structures.

The `TGNEW` program is a command line program. The syntax for the `TGNEW` command line is given in Figure 2.2. The square brackets, "[" and "]", indicate optional command line options and any associated values. Angle brackets, "<" and ">" indicate required values for the options when specified and required command line arguments. The ellipses, "...", are used to indicate that the line break is to be ignored, i.e., that the text on the second line should be typed on the same command line as the rest of the `TGNEW` command. The vertical bar, "|", indicates that there are

```
TGNEW [-h|-help]              ...
      [-ver|-version]         ...
      [-path <path>]          ...
      [-silent]               ...
      <tgdb name>             ...
      <schema file>           ...
      <species mapping file>
```

Figure 2.2: `TGNEW` command line syntax with the command line options and arguments.

synonyms for an option, either of which is valid.

All command line options must precede any command line arguments. The order of the command line options is not important, but options which have associated values must immediately precede their values. All command line arguments are required, and the ordering of the command line arguments is important. The three command line arguments for this program are the tree list generation database name, the tree list generation database schema filename, and the tree list generation database species mapping filename.

Options for the `TGNEW` program are indicated by a dash, or minus sign, immediately preceding the option name with no intervening spaces. Options may appear in any order but all options must appear before any command line arguments. Some options have associated values which must immediately follow the option separated from the option name by one or more blanks or spaces. The case of the command line option names is not important. The case of filenames, paths, etc., that are specified on the command line may be important depending on the operating system in use.

The complete list of the `TGNEW` program options are:

`-h` Display the online help message on the standard output, typically a terminal screen, and exit.

`-help` Display the online help message on the standard output, typically a terminal

screen, and exit.

`-silent` This option suppresses normal output to the standard output device. Error messages, if an error occurs while the program is running, are however, reported.

`-ver` Display the program version on the standard output, typically a terminal screen, and exit.

`-version` Display the program version on the standard output, typically a terminal screen, and exit.

`-path <path>` Define a tree list generation database path to be `<path>`. The tree list generation database path specifies the directory where a tree list generation database is located. If specified, the length of `<path>` may be limited, depending on the operating system being used. Each path element may be at most eight (8) characters in length and all of the path elements specified must exist. This option is not required.

Arguments for the `TGNEW` program are all required. The `TGNEW` command line arguments are:

`<tgdb name>` The name of the tree list generation database subdirectory that is to be created. This name is combined with the tree list generation database path, if specified, to generate a full path to the tree list generation database and its files and subdirectories. If the `-path` command line option is not specified, the tree list generation database is created in the current directory.

`<schema file>` The tree list generation database schema file. This file defines which treatments are to be included in a tree list generation database and which index parameters are to be used to index the tree measurement data. See Section 2.11.5 for details about the schema file.

```
PROMPT> tgnew sampledb schema.dat spmap.dat

Creating the tree list generation database: sampledb
in the current directory.
The tree list generation database was created successsfully.
```

Figure 2.3: Example of creating a new tree list generation database using `TGNEW`.

`<species map>` The tree list generation database species code to integer ID mapping file. This file defines unique two (2) and four to six (4-6) character species codes and unique integer identifiers for each tree species. See Section 2.11.4 for details about the species mapping file.

2.2.2 `TGNEW` example: Creating a tree list generation database

In this example, the `TGNEW` program is used with the tree list generation database species mapping file `spmap.dat` and the tree list generation database schema file `schema.dat` to create a tree list generation database `sampledb` in the current directory. Figure 2.3 presents the command used and the resulting output.

2.3 Adding measurement files to a tree list generation database

The tree list generation database program `TGADD` is used to add a stand measurement file, or a set of stand measurement files, to a tree list generation database. The required inputs for `TGADD` are the name of a stand measurement file and a tree list generation database name, when adding a single stand measurement file, or a stand measurement file list file name and a tree list generation database name, when adding a set of stand measurement files. The stand measurement file list file contains the name of each stand measurement file in the set of files to be added to a tree list generation database, one file name per line. The two ways in which `TGADD` may be used are indicated by the `-mf` and `-lf` command line options; the former indicating

that a single measurement file is to be added, and the latter indicating that a set of stand measurement files specified in a stand measurement file list file are to be added.

Be sure to create a backup of a tree list generation database before adding any stand measurement files. Stand measurement files are added tt a tree list generation database *live*, i.e., the new data are not staged before they are added to the database. This means that if an error occurs while adding stand measurement data with `TGADD`, the tree list generation database being modified will most likely become corrupted unusable.

2.3.1 The `TGADD` program

`TGADD` is a program that adds stand measurement data to a tree list generation database. Required inputs for this program are the tree list generation database name and a stand measurement file or a stand measurement file list file, as indicated by one of the option flags `-mf <file>` or `-lf <file>`, respectively. Only one of these options may be specified at a time. An optional input to the program is the name of a directory, or path, where the tree list generation database is located. Path elements are limited to eight (8) characters with filenames limited to an eight-dot-three format with eight (8) characters forming the base filename, followed by a dot or period, followed by a three (3) character filename extension. Filenames and path elements may not contain any blanks or other nonprinting characters. These common restrictions are imposed for the easy portability of filenames and directory structures.

The `TGADD` program is a command line program. The syntax for the `TGADD` command line when adding a single stand measurement file is given in Figure 2.4, and the syntax for the `TGADD` command line when adding a set of stand measurement files specified in a list file is given in Figure 2.5. The square brackets, "[" and "]", indicate optional command line options and their associated values. Angle brackets, "<" and ">" indicate required values for the options when specified and required command line arguments. Ellipses, "...", indicate that the new line is to be ignored, i.e., that

```
TGADD [-h|-help]              ...
      [-ver|-version]         ...
      [-path <path>]          ...
      [-silent]               ...
      [-max_tree <n>]         ...
      -mf <measurement file>  ...
      <tgdb name>
```

Figure 2.4: `TGADD` command line syntax with the command line options and arguments for adding a single stand measurement file to a tree list generation database.

```
TGADD [-h|-help]          ...
      [-ver|-version]     ...
      [-path <path>]      ...
      [-silent]           ...
      [-max_tree <n>]     ...
      -lf <listing file>  ...
      <tgdb name>
```

Figure 2.5: `TGADD` command line syntax with the command line options and arguments for adding a set of stand measurement files specified in a list file to a tree list generation database.

the text on the second line should be typed on the same command line as the rest of the `TGADD` command. The vertical bar, "|", indicates that there are synonyms for an option, either of which is valid. Before running `TGADD` to modify a tree list generation database, be sure to make a backup of the database. If an error occurs while adding data to a tree list generation database, the database could become corrupted. *Always* make a backup of a tree list generation database before adding new data.

All command line options must precede any command line arguments. The order of the command line options is not important, but options which have associated values must immediately precede their values. The only command line argument for this program is the tree list generation database name, and it is required.

Options for the `TGADD` program are indicated by a dash, or minus sign, immediately preceding the option name with no intervening spaces. Options may appear in any order but all options must appear before any command line arguments. Some options have associated values which must immediately follow the option separated from the option name by one or more blanks or spaces. The case of the command line option names is not important. The case of filenames, paths, etc., that are specified on the command line may be important depending on the operating system in use.

The list of command options for the `TGADD` program are:

`-h` Display the online help message on the standard output, typically a terminal screen, and exit.

`-help` Display the online help message on the standard output, typically a terminal screen, and exit.

`-silent` This option suppresses normal output to the standard output device. Error messages, if an error occurs while the program is running, are however, reported.

`-ver` Display the program version on the standard output, typically a terminal screen, and exit.

`-version` Display the program version on the standard output, typically a terminal screen, and exit.

`-path <path>` Define a tree list generation database path to be `<path>`. The tree list generation database path specifies the directory where a tree list generation database is located. If specified, the length of `<path>` may be limited, depending on the operating system being used. Each path element may be at most eight (8) characters in length and all of the path elements specified must exist. This option is not required.

`-mf <file>` Command line option indicating that the file named by the option value is a stand measurement file. A tree list generation database stand measurement file defines the stand treatment type, among other relevant stand attributes, and provides a list of tree measurements to be added to a tree list generation database. Tree measurements consist of a diameter at breast height measurement, a height measurement, and a two or four to six letter species code.

`-lf <file>` Command line option indicating that the file named by the option value is a listing file containing the names of stand measurement files whose data are to be added to a tree list generation database. The listing file provides an easy and efficient mechanism for adding multiple stand measurement files to a tree list generation database.

`-max_tree <n>` This option indicates that the program should allow a maximum of `<n>` trees when reading a stand measurement file. The number `<n>` must be a positive integer. The default value for the number of trees is 1024, which should be sufficient for most purposes. In the event that there are more than 1024 trees in a stand measurement file this option may be used to modify the maximum number of trees allowed so that the measurement file may be processed. This

option is not required.

Arguments for the TGADD program are all required. The TGADD command line argument is:

<tgdb name> The name of the tree list generation database to which tree measurement data from the stand measurement file, or files, are be added. This name is combined with the tree list generation database path, if specified, to generate a full path to a tree list generation database, and its files and directories. If the -path command line option is not specified, the tree list generation database specified is assumed to be in the current directory. It is important to create a backup of a tree list generation database before adding data; if any errors occur during the execution of the TGADD program the tree list generation database being modified may be corrupted. Always make a backup of the tree list generation database before adding new data.

2.3.2 TGADD examples: Adding data to a tree list generation database

In these examples the TGADD program is used to add stand measurement files to the tree list generation database sampledb created in the example for TGNEW in Section 2.2. The tree list generation database sampledb, the stand measurement files, and the stand measurement file list file are all assumed to be in the current directory. For these examples, there are eleven (11) stand measurement files that are to be added to the tree list generation database. The first example adds the stand measurement file file0001.dat. The second example uses the list file filelist.dat to add the remaining ten (10) files, file0002.dat through file0011.dat. The list file used for the second example may be found in Figure 2.8.

Figure 2.6 presents the command used and resulting output for the example which adds a single stand measurement file, file0001.dat, to the tree list generation database sampledb using the TGADD option -mf. Figure 2.7 presents the command

```
PROMPT> tgadd -mf file0001.dat sampledb

Processing measurement file: file0001.dat
```

Figure 2.6: Example of adding a single stand measurement file to a tree list generation database using `TGADD` and the `-mf` option.

```
PROMPT> tgadd -lf filelist.dat sampledb

Processing measurement file: file0002.dat
Processing measurement file: file0003.dat
Processing measurement file: file0004.dat
Processing measurement file: file0005.dat
Processing measurement file: file0006.dat
Processing measurement file: file0007.dat
Processing measurement file: file0008.dat
Processing measurement file: file0009.dat
Processing measurement file: file0010.dat
Processing measurement file: file0011.dat
```

Figure 2.7: Example of adding a set of stand measurement files to a tree list generation database using a list file and the `TGADD -lf` option.

used and resulting output for the example which adds multiple stand measurement files to the tree list generation database `sampledb` using a stand measurement file list file, `filelist.dat` and the `TGADD` option `-lf`.

2.4 Creating simulated tree lists using a tree list generation database

The tree list generation database program `TGRAND` is used to generate simulated tree lists, or random stand measurement files. `TGRAND` is the primary workhorse of the tree list generation database system. `TGRAND`requires a stand description file, or a set of stand description files specified in a list file, which describe the simulated tree list(s), or random stand measurement file(s), that are to be generated and a tree list generation database that is compatible with stand index parameter keywords used in

```
file0002.dat
file0003.dat
file0004.dat
file0005.dat
file0006.dat
file0007.dat
file0008.dat
file0009.dat
file0010.dat
file0011.dat
```

Figure 2.8: Stand measurement file listing file `filelist.dat` for `TGADD` example two, adding multiple stand measurement files to a tree list generation database using `TGADD` and the `-lf` option.

the stand description file or files. The stand description files used with a particular tree list generation database must contain appropriate values for the complete set of index parameters used by the tree list generation database for the desired treatment. See Section 2.11.2 for more information on stand description files and Sections 2.11.5 and 2.11.6 for more information on the specification and use of index parameter keywords in a tree list generation database.

`TGRAND` has four modes for generating simulated tree lists or random measurement files. The `TGRAND` stand generation mode is determined by the values of the `RANDOM_MF_PURE` and `RANDOM_MF_JITTER` keywords in a particular stand description file. Values for these keywords may be `YES`, indicating that the mode should be used, or `NO`, indicating that the mode should not be used. These two stand description file keywords and their values define the four `TGRAND` stand generation modes. Mnemonic acronyms for the four modes with the appropriate keyword values are given in table Table 2.3. The default mode for `TGRAND` is the RMF mode, which does not force 100% pure stands and does not jitter the simulated tree height values.

`TGRAND` uses a nearest neighbor selection procedure to determine which multidimensional histogram bins in a particular tree list generation database are similar to

Table 2.3: The four simulated tree list generation modes for the tree list generation database program TGRAND. Jittering tree heights adds or subtracts a small random amount to or from each height value. The 100% pure stand mode forces generated stands to be 100% of the trees to be the dominant species if they have a stand type of PURE_DF or PURE_WH. For details on how to set these properties for a tree list, see Section 2.11.2.

TGRAND mode	100% pure stands	Jitter heights
RMF	No	No
RMFJ	No	Yes
RMFP	Yes	No
RMFJP	Yes	Yes

a desired stand that is specified by a stand description file. If there are no data represented in a tree list generation database that are close enough to the desired stand, TGRAND will halt and give an error message indicating that there were no similar stands available in the database. As more data are added to a tree list generation database this issue will become less of a problem. There are two TGRAND options which control the maximum number of nearest neighbor multidimensional histogram bins that may be selected and used to generate a tree list and the maximum similarity score to be used in the nearest neighbor selection procedure. These are the -nn_max_stands option and the -nn_max_score option, respectively. The default value for the number of multidimensional histogram bins is 5 and the default value for the maximum nearest neighbor score is 5.0.

If TGRAND halts and indicates there were no multidimensional histogram bins near the desired stand description and the nearest neighbor score reported is greater than or equal to 10000, then there is no stand sufficiently near the desired stand. The reason for this is that the desired stand description differs in one of its categorical, or FIXED, type index parameters, e.g., STAND_TYPE, from all of the stands in a tree list generation database. There is nothing that may be done in this event to coerce TGRAND to generate a simulated tree list. The only way to resolve this problem is to add actual stand measurement data for the desired stand description to the tree list

generation database.

If TGRAND halts, indicating that there were no multidimensional histogram bins near the desired stand description and the nearest neighbor score reported is relatively small, e.g., less than 100.0, then a simulated tree list may be generated. The command line option -nn_max_score is used to increase the maximum nearest neighbor score until at least one multidimensional histogram bin is selected. A simulated tree list generated in this manner may or may not be representative of the desired stand description, and this should be verified before the tree list is used. This condition indicates that the *data space* spanned by the stands in a tree list generation database has small gaps, and these should be identified precisely so that data may be collected from temporary or permanent sample plots to fill the gaps. The TGSCATRP program and the scatterplot files it creates may prove useful for identifying gaps in a tree list generation database.

The simulated stands or tree lists produced by TGRAND and stored in a random stand measurement file do not contain tree multipliers to scale the number of trees to a hectare (acre) basis commonly used by growth and yield or stand simulation models [11, 20, 8, 35, 36]. Each tree in a simulated stand represents a single tree for the specified output plot size. Thus, if a hectare is the desired output plot size and the stand density has 1000 trees per hectare, the random measurement file will contain 1000 trees. If tree multipliers are necessary for compatibility with a particular growth and yield model, they may be obtained via the value of the PLOT_SIZE keyword in a random stand measurement file.

2.4.1 The TGRAND program

TGRAND is a program that generates a simulated tree list using a stand description file and a tree list generation database. Required inputs for this program are the tree list generation database name and a stand description file or a description file listing file, as indicated by one of the option flag -df <file> or -lf <file> respectively. Only

```
TGRAND [-h|-help]               ...
       [-ver|-version]          ...
       [-silent]                ...
       [-random_seed <seed>]    ...
       [-nn_max_score <score>]  ...
       [-nn_max_stands <n>]     ...
       [-path <path>]           ...
       -df <description file>   ...
       <tgdb name>
```

Figure 2.9: TGRAND command line syntax with the command line options and arguments for generating a single random stand measurement file from a stand description file using a tree list generation database.

one of these options may be specified at a time. An optional input to the program is the name of a directory where the tree list generation database is to be located. Path elements are limited to eight (8) characters with filenames limited to an eight-dot-three format with eight (8) characters forming the base filename, followed by a dot or period, followed by a three (3) character filename extension. Filenames and path elements may not contain any blanks or other nonprinting characters. These common restrictions are imposed for the easy portability of filenames and directory structures.

The TGRAND program is a command line program. The syntax for the TGRAND command line when generating a single random stand measurement file is given in Figure 2.9, and the syntax for the TGRAND command line when generating a set of random stand measurement files specified in a list file is given in Figure 2.10. The square brackets, "[" and "]", indicate optional command line options and their associated values. Angle brackets, "<" and ">" indicate required values for the options when specified and required command line arguments. Ellipses, "...", indicate that the new line is to be ignored, i.e., that the text on the second line should be typed on the same command line as the rest of the TGRAND command. The vertical bar, "|", indicates that there are synonyms for an option, either of which is valid.

```
TGRAND [-h|-help]               ...
       [-ver|-version]          ...
       [-silent]                ...
       [-random_seed <seed>]    ...
       [-nn_max_score <score>] ...
       [-nn_max_stands <n>]     ...
       [-path <path>]           ...
       -lf <listing file>       ...
       <tgdb name>
```

Figure 2.10: `TGRAND` command line syntax with the command line options and arguments for generating a set of random stand measurement files from a set of stand description files specified in a list file using a tree list generation database.

All command line options must precede any command line arguments. The order of the command line options is not important, but options which have associated values must immediately precede their values. The only command line argument for this program is the tree list generation database name, and it is required.

Options for the `TGRAND` program are indicated by a dash, or minus sign, immediately preceding the option name with no intervening spaces. Options may appear in any order but all options must appear before any command line arguments. Some options have associated values which must immediately follow the option separated from the option name by one or more blanks or spaces. The case of the command line option names is not important. The case of filenames, paths, etc., that are specified on the command line may be important depending on the operating system in use.

The list of options for the `TGRAND` program are:

`-h` Display the online help message on the standard output, typically a terminal screen, and exit.

`-help` Display the online help message on the standard output, typically a terminal screen, and exit.

`-silent` This option suppresses normal output to the standard output device. Error messages, if an error occurs while the program is running, are however, still reported.

`-ver` Display the program version on the standard output, typically a terminal screen, and exit.

`-version` Display the program version on the standard output, typically a terminal screen, and exit.

`-path <path>` Define a tree list generation database path to be `<path>`. The tree list generation database path specifies the directory where a tree list generation database is located. If specified, the length of `<path>` may be limited, depending on the operating system being used. Each path element may be at most eight (8) characters in length and all of the path elements specified must exist. This option is not required.

`-df <file>` Command line option indicating that the file named by the option value is a stand description file. A tree list generation stand description file defines the stand treatment type and other relevant stand defining attributes, allowing the generation of a random list of tree measurements appropriate for the stand description using a tree list generation database. The tree measurements consist of a diameter at breast height measurement, a height measurement, and a two or four to six letter species code. When using a single stand description file to generate a random stand measurement file, if an output file name is not specified using the `RANDOM_MF_NAME` keyword in the description file, the default filename of `output.rmf` will be used.

`-lf <file>` Command line option indicating that the file named by the option value is a listing file containing the names of stand description files that are to be used

with a tree list generation database to generate a random tree list. The listing file provides an easy and efficient mechanism for generating multiple random stand measurement files from a tree list generation database. When using a listing file, the names for all output random stand measurement files that are to be generated must be defined in the description files using the `RANDOM_MF_NAME` keyword.

`-random_seed <seed>` Command line option to specify a seed value for the uniform random number generator used to generate the simulated stands of trees. The seed should be a large, at least six (6) digits, relatively prime integer, but large and odd will generally suffice. This option allows the generation of multiple stands having similar characteristics, but with different simulated trees for different seed values when using the same description file. Alternatively, multiple description files for the same stand could be created and supplied to `TGRAND` via a listing file, using multiple copies of the same description file with different output filenames, to create similar stands with different sets simulated trees. This latter method is the preferred method for generating multiple stands having similar characteristics since a complete record of the stands generated and their descriptions are contained in the random measurement files and the description files. This option is not required.

`-nn_max_score <score>` Command line option which sets the maximum value for the nearest neighbor similarity score to be used when selecting the tree measurement data from the tree list generation database that are used to generate a simulated or random stand of trees. The score values are somewhat arbitrary, but are based on a similarity index. A score of zero would indicate an exact match between the attributes of a desired stand and the attributes for a stand whose data are contained in the tree list generation database. A default maximum similarity score of 5.0 is used. By default, only the best 5 multidimensional

histogram bins, and their associated tree data, as determined by their scores, are used to generate a simulated stand. This default may be overridden by using the `-nn_max_stands` option. This option is not required.

`-nn_max_stands <n>` Command line option which sets the maximum value for the number of neighboring stands, multidimensional histogram bins, from the tree list generation database that are to be used when generating a simulated stand of trees. This option may be used in conjunction with the `-nn_max_score` option to control the amount of data pooled to create a canonical stand from which a simulated tree list is generated. The default value for this option is 5 stands. This option is not required.

The single argument for the `TGRAND` program is required. The `TGRAND` command line argument is:

`<tgdb name>` The name of the tree list generation database to which tree measurement data from the stand measurement file, or files, are be added. This name is combined with the tree list generation database path, if specified, to generate a full path to a tree list generation database, and its files and directories. If the `-path` command line option is not specified, the tree list generation database specified is assumed to be in the current directory.

2.4.2 TGRAND examples: Generating simulated stands or tree lists

In these examples the `TGRAND` program is used to generate random stand measurement files from the tree list generation database `tgdb1r00`. The tree list generation database `tgdb1r00` and the stand description files used are all assumed to be in the current directory. `TGRAND` is used to create a random stand measurement file for an untreated stand and a random stand measurement file for a thinned stand. An example using `TGRAND` with a stand description file list file is not presented. The `-lf` option of

```
PROMPT> tgrand -df untreatd.df tgdb1r00

Processing description file: untreatd.df
   Creating random measurement file: untreatd.rmf
```

Figure 2.11: Example of generating a single random stand measurement file for an untreated stand specified by the file `untreatd.df`, Figure 2.12, from a tree list generation database using `TGRAND`.

`TGRAND` and the list file format are exactly the same as the `-lf` option and list file format of `TGADD`.

Figure 2.11 presents the command used and resulting output for the example which creates a single random stand measurement file for an untreated stand using the description file `untreatd.df` and the tree list generation database `tgdb1r00`. Figure 2.12 presents the stand description file used, and Figure 2.13 presents height *vs.* diameter plots for actual and simulated tree lists for the four `TGRAND` stand generation modes for the untreated stand example.

Figure 2.14 presents the command used and resulting output for the example which creates a single random stand measurement file for a thinned stand using the description file `thinned.df` and the tree list generation database `tgdb1r00`. Figure 2.15 presents the stand description file used, and Figure 2.16 presents height *vs.* diameter plots for actual and simulated tree lists for the four `TGRAND` stand generation modes for the thinned stand example.

2.5 Summarizing a tree list generation database

The tree list generation database program `TGSUMRY` is used to create a brief summary of the contents and status of a tree list generation database. The summary information includes the number of treatments in a tree list generation database, the index parameters used for each treatment, whether the tree list generation database

```
%
%  Tree list generator stand description file.
%
%  File created: 11/08/1999 at 14:19:50.14
%
MEASUREMENT_UNITS              = METRIC
TREATMENT_TYPE                 = UNTREATED
SITE_INDEX_50                  = 35.966
STAND_TOTAL_AGE                = 27.000
STAND_ORIGIN                   = NATURAL
QMD                            = 14.277
STAND_DENSITY                  = 3951.000
STAND_TYPE                     = WH_DOMINANT
RANDOM_MF_UNITS                = METRIC
RANDOM_MF_PLOT_SIZE            = 0.041
RANDOM_MF_PURE                 = NO
RANDOM_MF_JITTER               = NO
RANDOM_MF_NAME                 = untreatd.rmf
```

Figure 2.12: Stand description file `untreatd.df` for `TGRAND` example one, generating a simulated tree list for an untreated stand from a tree list generation database using `TGRAND` and the `-df` option. 162 trees are generated for this stand description file.

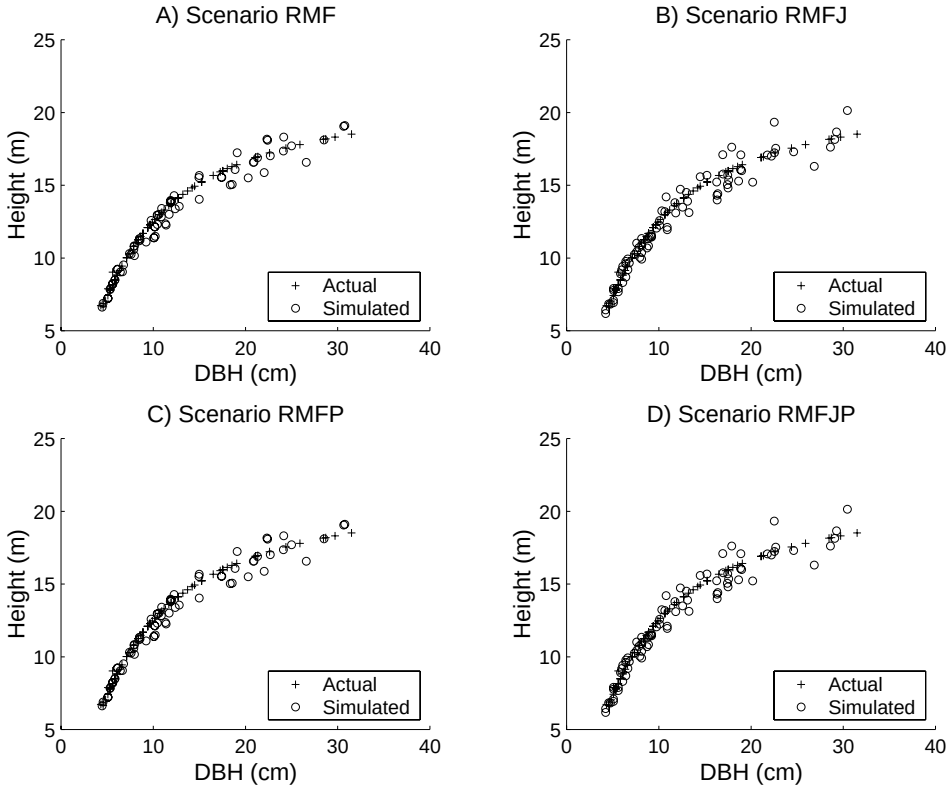

Figure 2.13: Height *vs.* diameter plots for an untreated stand. The plots show data for the actual stand represented by the stand description file `untreatd.df` and simulated tree lists generated by `TGRAND` for each of the four stand generation modes. Only half the trees sampled for this plot were plotted in the figures.

```
PROMPT> tgrand -df thinned.df tgdb1r00

Processing description file: thinned.df
   Creating random measurement file: thinned.rmf
```

Figure 2.14: Example of generating a single random stand measurement file for a thinned stand specified by the file `untreatd.df`, Figure 2.15, from a tree list generation database using `TGRAND`.

```
%
%  Tree list generator stand description file.
%
%  File created: 11/03/1999 at 16:01:34.24
%
MEASUREMENT_UNITS              = IMPERIAL
TREATMENT_TYPE                 = THINNED
SITE_INDEX_50                  = 129.900
STAND_TOTAL_AGE                = 76.000
STAND_ORIGIN                   = NATURAL
NUMBER_OF_THINNINGS            = 1.000
PRE_THIN_DENSITY               = 437.000
PRE_THIN_BASAL_AREA            = 242.200
PCT_BASAL_AREA_REMOVED         = 34.600
YEARS_SINCE_TREATMENT          = 2.900
QMD                            = 11.425
STAND_DENSITY                  = 243.000
STAND_TYPE                     = PURE_DF
RANDOM_MF_UNITS                = IMPERIAL
RANDOM_MF_PLOT_SIZE            = 0.124
RANDOM_MF_PURE                 = NO
RANDOM_MF_JITTER               = NO
RANDOM_MF_NAME                 = thinned.rmf
```

Figure 2.15: Stand description file `thinned.df` for `TGRAND` example two, generating a simulated tree list for a thinned stand from a tree list generation database using `TGRAND` and the `-df` option. 29 trees are generated for this stand description file.

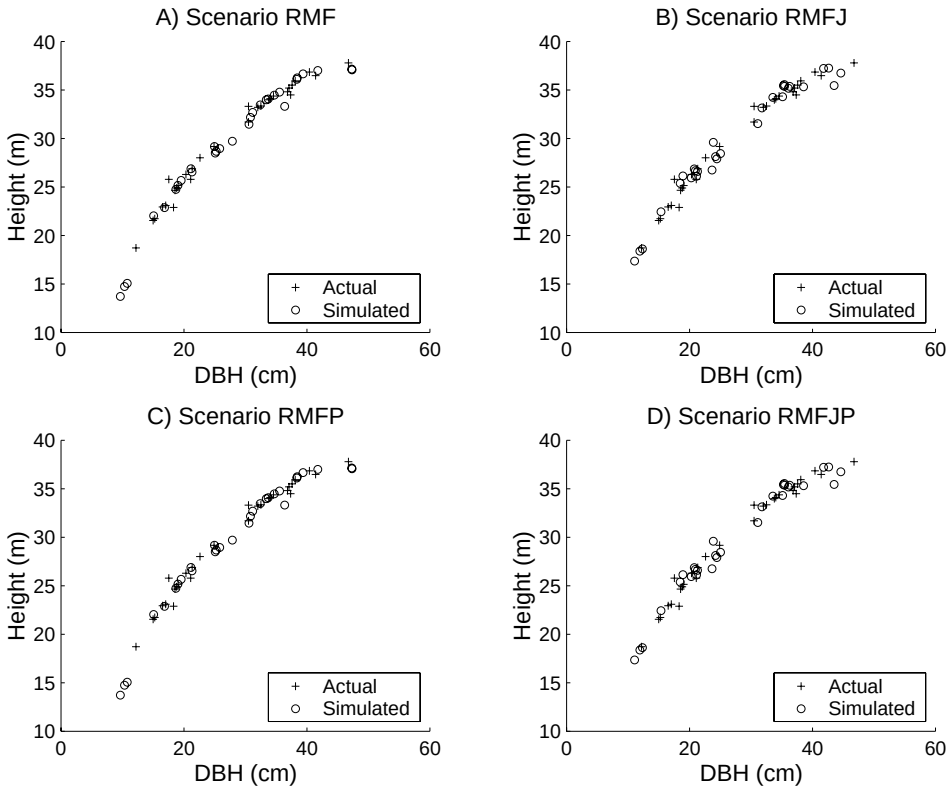

Figure 2.16: Height *vs.* diameter plots for a thinned stand. The plots show data for the actual stand represented by the stand description file `thinned.df` and simulated tree lists generated by `TGRAND` for each of the four stand generation modes.

```
TGSUMRY [-h|-help]              ...
        [-ver|-version]         ...
        [-silent]               ...
        [-path <path>]          ...
        [-file <file>]          ...
        <tgdb name>
```

Figure 2.17: `TGSUMRY` command line syntax with the command line options and arguments.

is locked or unlocked, and a variety of other potentially useful items. `TGSUMRY` will save the summary information to an output file, specified using the `-file` option, or display the summary information on the standard output device, generally a terminal screen or shell window. The output from the `TGSUMRY` program for the tree list generation database `tgdb1r00` is provided in Appendix A.

2.5.1 The `TGSUMRY` program

`TGSUMRY` is a program that displays a brief summary of a tree list generation database. The only required input for this program is the tree list generation database name. An optional input to the program is the name of a directory, or path, where a tree list generation database is located. Path elements are limited to eight (8) characters with filenames limited to an eight-dot-three format with eight (8) characters forming the base filename, followed by a dot or period, followed by a three (3) character filename extension. Filenames and path elements may not contain any blanks or other nonprinting characters. These common restrictions are imposed for the easy portability of filenames and directory structures.

The `TGSUMRY` program is a command line program. The syntax for the `TGSUMRY` command line is given in Figure 2.17. The square brackets, "[" and "]", indicate optional command line options and their associated values. Angle brackets, "<" and ">" indicate required values for the options when specified and required command line arguments. The ellipses, "...", are used to indicate that the line break is to be

ignored, i.e., that the text on the second line should be typed on the same line as the rest of the `TGSUMRY` command. The vertical bar, "|", indicates that there are synonyms for an option, either of which is valid.

All command line options must precede any command line arguments. The order of the command line options is not important, but options which have associated values must immediately precede their values. The only command line argument for this program is the tree list generation database name, and it is required.

Options for the `TGSUMRY` program are indicated by a dash, or minus sign, immediately preceding the option name with no intervening spaces. Options may appear in any order but all options must appear before any command line arguments. Some options have associated values which must immediately follow the option separated from the option name by one or more blanks or spaces. The case of the command line option names is not important. The case of filenames, paths, etc., that are specified on the command line may be important depending on the operating system in use.

The list of options for the `TGSUMRY` program are:

`-h` Display the online help message on the standard output, typically a terminal screen, and exit.

`-help` Display the online help message on the standard output, typically a terminal screen, and exit.

`-silent` This option suppresses normal output to the standard output device. Error messages, if an error occurs while the program is running, are however, reported.

`-ver` Display the program version on the standard output, typically a terminal screen, and exit.

`-version` Display the program version on the standard output, typically a terminal screen, and exit.

`-file <file>` This option specifies a file to which the tree list generation summary information is to be written. The filename specified must be the name of a new file. This option is not required. If this option is not specified, the summary information will be displayed on the standard output, typically a terminal screen.

`-path <path>` Define a tree list generation database path to be `<path>`. The tree list generation database path specifies the directory where a tree list generation database is located. If specified, the length of `<path>` may be limited, depending on the operating system being used. Each path element may be at most eight (8) characters in length and all of the path elements specified must exist. This option is not required.

The argument for the `TGSUMRY` program is required. The `TGSUMRY` command line argument is:

`<tgdb name>` The name of the tree list generation database that is to be summarized by `TGSUMRY`. This name is combined with the tree list generation database path, if specified, to generate a full path to a tree list generation database, and its files and directories. If the `-path` command line option is not specified, the tree list generation database specified is assumed to be in the current directory.

2.5.2 `TGSUMRY` example: Summarizing a tree list generation database

In this example, the `TGSUMRY` program is used to generate a summary for the tree list generation database `tgdb1r00`, which is in the current directory. The `TGSUMRY` option `-file` is used to place the tree list generation database summary into the file `summary.txt`, also in the current directory. The tree list generation database summary identifies which treatments are in the database, which index parameters are used for each treatment, and some other useful information, such as the time and date

```
PROMPT> tgsumry -file summary.txt tgdb1r00

Writing information to file: summary.txt
```

Figure 2.18: Example summarizing a tree list generation database using `TGSUMRY`. See Appendix A for the contents of the file `summary.txt`.

a tree list generation database was last modified. Figure 2.18 presents the command used and resulting output. See Appendix A for the contents of the file `summary.txt`.

2.6 Locking and unlocking a tree list generation database

The tree list generation database programs `TGLOCK` and `TGUNLOCK` are used to prevent or permit the addition of new stand measurement data to a tree list generation database. The name of the tree list generation database that is to be locked or unlocked is the only required input to the programs. The `TGLOCK` and `TGUNLOCK` programs may be used to restrict the ability to add data to a tree list generation database to a selected group within an organization, e.g., the group which collects and conditions the stand measurement data before it is distributed to the users of the data. The average user of a tree list generation database should not necessarily be adding data.

2.6.1 The `TGLOCK` program

`TGLOCK` is a program that locks a tree list generation database preventing the addition of tree measurement data or other modification of the database. Once locked, a tree list generation database may be unlocked using the program `TGUNLOCK`, permitting the addition of new tree measurement data.

The only required input for this program is the tree list generation database name. An optional input to the program is the name of a directory, or path, where a tree list generation database is located. Path elements are limited to eight (8) characters

```
TGLOCK [-h|-help]              ...
       [-ver|-version]         ...
       [-path <path>]          ...
       [-silent]               ...
       <tgdb name>
```

Figure 2.19: TGLOCK command line syntax with the command line options and arguments.

with filenames limited to an eight-dot-three format with eight (8) characters forming the base filename, followed by a dot or period, followed by a three (3) character filename extension. Filenames and path elements may not contain any blanks or other nonprinting characters. These common restrictions are imposed for the easy portability of filenames and directory structures.

The TGLOCK program is a command line program. The syntax for the TGLOCK command line is given in Figure 2.19. The square brackets, "[" and "]", indicate optional command line options and their associated values. Angle brackets, "<" and ">" indicate required values for the options when specified and required command line arguments. Ellipses, "...", indicate that the new line is to be ignored, i.e., that the text on the second line should be typed on the same line as the rest of the TGLOCK command. The vertical bar, "|", indicates that there are synonyms for an option, either of which is valid.

All command line options must precede any command line arguments. The order of the command line options is not important, but options which have associated values must immediately precede their values. The only command line argument for this program is the tree list generation database name, and it is required.

Options for the TGLOCK program are indicated by a dash, or minus sign, immediately preceding the option name with no intervening spaces. Options may appear in any order but all options must appear before any command line arguments. Some options have associated values which must immediately follow the option separated

from the option name by one or more blanks or spaces. The case of the command line option names is not important. The case of filenames, paths, etc., that are specified on the command line may be important depending on the operating system in use.

The list of options for the `TGLOCK` program are:

`-h` Display the online help message on the standard output, typically a terminal screen, and exit.

`-help` Display the online help message on the standard output, typically a terminal screen, and exit.

`-silent` This option suppresses normal output to the standard output device. Error messages, if an error occurs while the program is running, are however, reported.

`-ver` Display the program version on the standard output, typically a terminal screen, and exit.

`-version` Display the program version on the standard output, typically a terminal screen, and exit.

`-path <path>` Define a tree list generation database path to be `<path>`. The tree list generation database path specifies the directory where a tree list generation database is located. If specified, the length of `<path>` may be limited, depending on the operating system being used. Each path element may be at most eight (8) characters in length and all of the path elements specified must exist. This option is not required.

The argument for the `TGLOCK` program is required. The `TGLOCK` command line argument is:

```
PROMPT> tglock sampledb

The tree list generation database is now locked.
```

Figure 2.20: Example of locking a tree list generation database using TGLOCK.

<tgdb name> The name of the tree list generation database that is to be locked, preventing further modification until the tree list generation database is unlocked using TGUNLOCK. This name is combined with the tree list generation database path, if specified, to generate a full path to a tree list generation database, and its files and directories. If the -path command line option is not specified, the tree list generation database specified is assumed to be in the current directory.

2.6.2 TGLOCK example: Locking a tree list generation database

In this example, the TGLOCK program is used to lock the tree list generation database sampledb in the current directory. This prevents further modification of the tree list generation database, and does not allow the addition of new stand measurement files. Figure 2.20 presents the command used and resulting output.

2.6.3 The TGUNLOCK program

TGUNLOCK is a program that unlocks a tree list generation database allowing the addition of tree measurement data or other modification of a database. Once unlocked, a tree list generation database may be locked using the program TGLOCK, preventing the addition of tree measurement or other modification.

The only required input for this program is the tree list generation database name. An optional input to the program is the name of a directory, or path, where a tree list generation database is located. Path elements are limited to eight (8) characters with filenames limited to an eight-dot-three format with eight (8) characters forming

```
TGUNLOCK [-h|-help]              ...
         [-ver|-version]         ...
         [-path <path>]          ...
         [-silent]               ...
         <tgdb name>
```

Figure 2.21: TGUNLOCK command line syntax with the command line options and arguments.

the base filename, followed by a dot or period, followed by a three (3) character filename extension. Filenames and path elements may not contain any blanks or other nonprinting characters. These common restrictions are imposed for the easy portability of filenames and directory structures.

The TGUNLOCK program is a command line program. The syntax for the TGUNLOCK command line is given in Figure 2.21. The square brackets, "[" and "]", indicate optional command line options and their associated values. Angle brackets, "<" and ">" indicate required values for the options when specified and required command line arguments. Ellipses, "...", indicate that the new line is to be ignored, i.e., that the text on the second line should be typed on the same line as the rest of the TGUNLOCKcommand. The vertical bar, "|", indicates that there are synonyms for an option, either of which is valid.

All command line options must precede any command line arguments. The order of the command line options is not important, but options which have associated values must immediately precede their values. The only command line argument for this program is the tree list generation database name, and it is required.

Options for the TGUNLOCKprogram are indicated by a dash, or minus sign, immediately preceding the option name with no intervening spaces. Options may appear in any order but all options must appear before any command line arguments. Some options have associated values which must immediately follow the option separated from the option name by one or more blanks or spaces. The case of the command line

option names is not important. The case of filenames, paths, etc., that are specified on the command line may be important depending on the operating system in use.

The list of options for the TGUNLOCKprogram are:

`-h` Display the online help message on the standard output, typically a terminal screen, and exit.

`-help` Display the online help message on the standard output, typically a terminal screen, and exit.

`-silent` This option suppresses normal output to the standard output device. Error messages, if an error occurs while the program is running, are however, reported.

`-ver` Display the program version on the standard output, typically a terminal screen, and exit.

`-version` Display the program version on the standard output, typically a terminal screen, and exit.

`-path <path>` Define a tree list generation database path to be `<path>`. The tree list generation database path specifies the directory where a tree list generation database is located. If specified, the length of `<path>` may be limited, depending on the operating system being used. Each path element may be at most eight (8) characters in length and all of the path elements specified must exist. This option is not required.

The argument for the TGUNLOCKprogram is required. The TGUNLOCKcommand line argument is:

`<tgdb name>` The name of the tree list generation database that is to be unlocked, permitting modifications until the tree list generation database is locked using

```
PROMPT> tgunlock sampledb

The tree list generation database is now unlocked.
```

Figure 2.22: Example of unlocking a tree list generation database using TGUNLOCK.

TGLOCK. This name is combined with the tree list generation database path, if specified, to generate a full path to a tree list generation database, and its files and directories. If the "-path" command line option is not specified, the tree list generation database specified is assumed to be in the current directory.

2.6.4 TGUNLOCK example: Unlocking a tree list generation database

In this example, the TGUNLOCK program is used to unlock the locked tree list generation database sampledb in the current directory. This allows modification of the tree list generation database and the addition of new stand measurement files. Figure 2.22 presents the command used and resulting output.

2.7 Exploding a tree list generation database

The tree list generation database program TGXPLODE is used to convert a specified tree list generation database into its component files, the schema file, the species mapping file, and the individual stand measurement files for each treatment. TGXPLODE requires the name of the tree list generation database that is to be converted into its components and the name of a destination directory for the files. The file and directory structure for the exploded tree list generation database is identical to that of a normal tree list generation database, except the individual stand measurement files will replace the tree list generation database files for each treatment. The schema file, the species mapping file, and the treatment subdirectories are placed in the destination directory. Descriptions of the schema file, the species mapping file, and the

treatment measurement files may be found in Section 2.11.5 and Section 2.11.4, and Section 2.11.1, respectively. For a description of the tree list generation database file and directory structure, see Section 2.1.2.

`TGXPLODE` may be used in conjunction with `TGNEW` and `TGADD` to reconfigure the treatments in a tree list generation database, and the stand index parameters and their multidimensional histogram based classification of the stand measurement data in a tree list generation database. This may done in three steps. First, the tree list generation database schema files is modified to reflect the desired changes in index parameters of histogram bin widths. Second, `TGNEW` is used to create a new tree list generation database. Third, `TGADD` is used to add all of the stand measurement files into the reconfigured tree list generation database.

Two scenarios for the application of this procedure readily present themselves. First, if a tree list generation database has a very sparse classification, caused by using too many index parameters or having histogram bin widths that are too narrow, the number of index parameters used and the histogram bin widths may be modified to resolve this problem. Second, the multidimensional histogram based classification procedure used to implement the tree list generation database permits the refinement, or other modification, of the histogram bin widths as more data are added to a database.

2.7.1 The `TGXPLODE` program

`TGXPLODE` is a program that extracts the individual source stand measurement files, schema file, and species mapping file from a tree list generation database. `TGXPLODE` extracts this information from a specified tree list generation database and places the individual files into a specified destination directory and treatment subdirectories of the destination directory. This program allows the data buckets in a tree list generation database to be refined, i.e., it allows the creation of a new tree list generation database with smaller, or different interval widths or index parameter sets

```
TGXPLODE [-h|-help]              ...
         [-ver|-version]         ...
         [-silent]               ...
         [-path <path>]          ...
         <tgdb name>             ...
         <destination dir>
```

Figure 2.23: TGXPLODE command line syntax with the command line options and arguments.

from an existing tree list generation database. For example, the original tree list generation database could eventually contain enough data to add more index parameters or smaller interval widths, both providing a refinement of the database, creating more data buckets, and allowing the generation of random stands which more closely approximate the desired stands.

The required input arguments for this program are the tree list generation database name and the destination directory or path. An optional input to the program is the name of a directory, or path, where a tree list generation database is located. Path elements are limited to eight (8) characters with filenames limited to an eight-dot-three format with eight (8) characters forming the base filename, followed by a dot or period, followed by a three (3) character filename extension. Filenames and path elements may not contain any blanks or other nonprinting characters. These common restrictions are imposed for the easy portability of filenames and directory structures.

The TGXPLODE program is a command line program. The syntax for the TGXPLODE command line is given in Figure 2.23. The square brackets, "[" and "]", indicate optional command line options and their associated values. Angle brackets, "<" and ">" indicate required values for the options when specified and required command line arguments. The ellipses, "...", are used to indicate that the line break is to be ignored, i.e., that the text on the second line should be typed on the same line as the rest of the TGSUMRY command. The vertical bar, "|", indicates that there are synonyms for an option, either of which is valid.

All command line options must precede any command line arguments. The order of the command line options is not important, but options which have associated values must immediately precede their values. All command line arguments are required, and the ordering of the command line arguments when several arguments are present is important. The two command line arguments for this program are the tree list generation database name and the destination directory for the database component files.

Options for the `TGXPLODE` program are indicated by a dash, or minus sign, immediately preceding the option name with no intervening spaces. Options may appear in any order but all options must appear before any command line arguments. Some options have associated values which must immediately follow the option separated from the option name by one or more blanks or spaces. The case of the command line option names is not important. The case of filenames, paths, etc., that are specified on the command line may be important depending on the operating system in use.

The list of options for the `TGXPLODE` program are:

`-h` Display the online help message on the standard output, typically a terminal screen, and exit.

`-help` Display the online help message on the standard output, typically a terminal screen, and exit.

`-silent` This option suppresses normal output to the standard output device. Error messages, if an error occurs while the program is running, are however, reported.

`-ver` Display the program version on the standard output, typically a terminal screen, and exit.

`-version` Display the program version on the standard output, typically a terminal screen, and exit.

`-path <path>` Define a tree list generation database path to be `<path>`. The tree list generation database path specifies the directory where a tree list generation database is located. If specified, the length of `<path>` may be limited, depending on the operating system being used. Each path element may be at most eight (8) characters in length and all of the path elements specified must exist. This option is not required.

The arguments for the `TGXPLODE` program are required. The `TGXPLODE` command line arguments are:

`<tgdb name>` The name of the tree list generation database that is to be converted into its component parts by `TGXPLODE`. This name is combined with the tree list generation database path, if specified, to generate a full path to a tree list generation database, and its files and directories. If the `-path` command line option is not specified, the tree list generation database specified is assumed to be in the current directory.

`<destination dir>` The destination directory for the component files of the specified tree list generation database `<tgdb name>`. This directory must exist. The destination directory "." may be used to indicate that the current directory is the destination. The schema file and the species mapping file for the specified tree list generation database, `schema.dat` and `spmap.dat`, respectively, are both written to the destination directory. Additionally, for each treatment defined in the specified tree list generation database, a treatment directory is created in the destination directory and stand measurement files containing the treatment data placed into the tree list generation database created in the appropriate treatment subdirectory of the destination directory.

```
PROMPT> tgxplode tgdb1r00 kaboom

Exploding the tree list generation database: tgdb1r00
Located in current directory
To the destination directory: kaboom

PROMPT> dir kaboom

schema.dat
spmap.dat
t
u
```

Figure 2.24: Example of exploding a tree list generation database into its component files using TGXPLODE. The component files are a schema file, a species mapping file, and individual stand measurement files for each treatment in their appropriate subdirectory.

2.7.2 TGXPLODE example: Exploding a tree list generation database

In this example the TGXPLODE program is used to convert the tree the tree list generation database tgdb1r00 into its component parts: a schema file, a species mapping file, and the measurement files for each treatment. The tree list generation database is assumed to be in the current directory and the destination directory is named kaboom. Figure 2.24 presents the command used and resulting output. Also shown in the figure is a directory listing of the destination directory kaboom. The treatment subdirectories kaboom/u and kaboom/t contain the individual stand measurement files for the two treatments.

2.8 Creating a scatterplot file from a tree list generation database

The tree list generation database program TGSCATRP is used to create a scatterplot file for a specified tree list generation database and treatment. The scatterplot file is a comma delimited text file with the index parameters for the specified treatment as

the data column headings, and the centers of the multidimensional histogram bins, or data buckets, as the data. The scatterplot file may be used with other software to visually assess the data coverage in a tree list generation database to determine treatment combinations or other stand conditions that are not represented. The output of TGSCATRP when combined with the output of TGSUMRY gives a more or less complete description of the contents of a tree list generation database.

2.8.1 The TGSCATRP program

TGSCATRP is a program that extracts the centers of the multidimensional histogram bins, or data buckets, from the index of tree list generation database. TGSCATRP extracts the index parameter names and data from a specified tree list generation database and creates comma delimited output that is displayed on the standard output device, typically a terminal screen, or written to a specified output file. The scatter plot information may be used to determine whether there are gaps in the data contained in a tree list generation database by plotting one index data column against another. For example, by plotting site index against stand age one could determine whether there are particular combinations of these two index parameters that are not in the database, indicating protocols for stands that ought to be sampled in the future.

The required inputs for this program are the tree list generation database name and the name of the treatment whose scatterplot data are desired. An optional input to the program is the name of a directory, or path, where a tree list generation database is located and an output file to which the scatterplot data are to be written. Path elements are limited to eight (8) characters with filenames limited to an eight-dot-three format with eight (8) characters forming the base filename, followed by a dot or period, followed by a three (3) character filename extension. Filenames and path elements may not contain any blanks or other nonprinting characters. These common restrictions are imposed for the easy portability of file names and directory

```
TGSCATRP [-h|-help]             ...
         [-ver|-version]         ...
         [-silent]               ...
         [-path <path>]          ...
         [-file <file>]          ...
         -treatment <treatment> ...
         <tgdb name>
```

Figure 2.25: TGSCATRP command line syntax with the command line options and arguments.

structures.

The TGSCATRP program is a command line program. The syntax for the TGSCATRP command line is given in Figure 2.25. The square brackets, "[" and "]", indicate optional command line options and their associated values. Angle brackets, "<" and ">" indicate required values for the options when specified and required command line arguments. The ellipses, "...", are used to indicate that the line break is to be ignored, i.e., that the text on the second line should be typed on the same line as the rest of the TGSUMRY command. The vertical bar, "|", indicates that there are synonyms for an option, either of which is valid.

All command line options must precede any command line arguments. The order of the command line options is not important, but options which have associated values must immediately precede their values. The only command line argument for this program is the tree list generation database name, and it is required.

Options for the TGSCATRP program are indicated by a dash, or minus sign, immediately preceding the option name with no intervening spaces. Options may appear in any order but all options must appear before any command line arguments. Some options have associated values which must immediately follow the option separated from the option name by one or more blanks or spaces. The case of the command line option names is not important. The case of filenames, paths, etc., that are specified on the command line may be important depending on the operating system in use.

The list of options for the `TGSCATRP` program are:

`-h` Display the online help message on the standard output, typically a terminal screen, and exit.

`-help` Display the online help message on the standard output, typically a terminal screen, and exit.

`-silent` This option suppresses normal output to the standard output device. Error messages, if an error occurs while the program is running, are however, reported.

`-ver` Display the program version on the standard output, typically a terminal screen, and exit.

`-version` Display the program version on the standard output, typically a terminal screen, and exit.

`-path <path>` Define a tree list generation database path to be `<path>`. The tree list generation database path specifies the directory where a tree list generation database is located. If specified, the length of `<path>` may be limited, depending on the operating system being used. Each path element may be at most eight (8) characters in length and all of the path elements specified must exist. This option is not required.

`-file <file>` This option specifies a file to which the tree list generation scatter plot information is to be written. The filename specified must be the name of a new file. This option is not required. If not specified, the scatter plot information will be written to the standard output device, typically a terminal screen.

`-treatment <treatment>` This option specifies the name of the treatment whose scatterplot data are desired. It must be one of the treatments contained in the

```
PROMPT> tgscatrp -file untreatd.dat -treatment untreated tgdb1r00

Writing scatter plot data to file: untreatd.dat
```

Figure 2.26: Example of creating a tree list generation database scatterplot file for the untreated stand data using TGSCATRP.

named tree list generation database. This option is required.

The argument for the TGSCATRP program is required. The TGSCATRP command line argument is:

<tgdb name> The name of the tree list generation database for which a comma delimited treatment scatter plot file by TGSCATRP. This name is combined with the tree list generation database path, if specified, to generate full path to a tree list generation database, and its files and directories. If the -path command line option is not specified, the tree list generation database specified is assumed to be in the current directory.

2.8.2 TGSCATRP examples: Creating scatterplot files

In these examples the TGSCATRP program is used to create scatterplot data files for the untreated stand data and the thinned stand data in the tree list generation database tgdb1r00. The tree list generation database is assumed to be in the current directory. Figure 2.26 and Figure 2.27 present the command used and resulting output. The contents of the untreated and thinned stand scatterplot files is not presented directly due to its large volume, but Section 2.10 presents a set of figures derived from the scatterplot data indicating the coverage of the data in the tree list generation database tgdb1r00.

```
PROMPT> tgscatrp -file thinned.dat -treatment thinned tgdb1r00

Writing scatter plot data to file: thinned.dat
```

Figure 2.27: Example of creating a tree list generation database scatterplot file for the thinned stand data using `TGSCATRP`.

2.9 Tree list generation database software error messages

The tree list generation database software has three levels of error messages that may occur. All errors that are detected by the tree list generation database software, regardless of level, are considered to be fatal errors, that is, they halt the execution of the program that was running when the error occurred. The three levels correspond to the point at which an error of some sort is first detected and reported. The tree list generation database software error messages report problems that occur at the level of the program, e.g., `TGRAND`, which is the highest level, at the level of the tree list generation database subroutine library, which is the middle level, and at the level of the subroutine library used to build the tree list generation database subroutine library, the lowest level.

When an error occurs, an error message is displayed on the standard output device and program execution halts. The error message contains a variety of information intended to be helpful in resolving the problem which caused the error occur. First, an error message contains a toolkit version, which refers to the version of the SPICE system and its subroutine library, developed and maintained by the Navigation and Ancillary Information Facility (NAIF) of NASA's Jet Propulsion Laboratory, which were used to implement the tree list generation database subroutine library. See Appendix F for a brief description of the SPICE system and NAIF. Second, an error message contains a short error indicator that is intended to identify the source of the error and give a mnemonic for the type of error. This part of the error message is

intended for processing by a computer. Third, an error message contains a description of the error that occurred and the relevant context, e.g., missing values, invalid values, etc. This part of the error message is intended to help determine exactly what caused the error to occur. Finally, an error message contains traceback information giving the list of modules that were called before the error was detected. This last information is likely to be of use only to software developers working with the tree list generation database subroutine library, and may be usually ignored by the more typical users of the tree list generation database system.

```
==========================================================================

Toolkit version: N0048

<level>(<error indicator>)

<error description>

A traceback follows.  The name of the highest level module is first.
<program> --> <subroutine 1> -->  ... <subroutine N>

==========================================================================
```

Figure 2.28: Tree list generation database error message format.

Figure 2.28 provides a template for the format of the tree list generation database error messages. The toolkit version, `Toolkit version: N0048` indicates the version of the SPICE toolkit used to develop the tree list generation database software. This will remain the same until, or unless, a more recent version of the SPICE toolkit is obtained and incorporated into the tree list generation database system. The `<level>` tag indicates the level of the software where the error occurred. In an actual error message the `<level>` tag will be replaced with the program name, e.g., `TGADD`, if the error occurs within a program, or with `TGLIB` if the error occurs in the tree list generation database subroutine library, or with `SPICE` if the error occurs in the SPICE subroutine library. The `<error indicator>` tag gives a short mnemonic for

an error that occurs, e.g., USAGEERROR or NOSUCHFILE. The <error description> tag provides a text description of the type of error that occurred and possibly some additional context information such as the valid range for a value. This is the part of the error message that should provide the most help in determining the cause of an error. Finally, there is the module traceback which gives the list of module names in the order that they were called before the error occurred. Figure 2.29 provides an example of a program level error message using TGRAND.

```
PROMPT> TGRAND

======================================================================

Toolkit version: N0048

TGRAND(USAGEERROR)

Empty command line. Type 'TGRAND -h' for help.

A traceback follows.  The name of the highest level module is first.
TGRAND

======================================================================
```

Figure 2.29: Example error message from TGRAND. In this example, TGRAND was invoked without any command line arguments, resulting in the error message displayed.

2.10 Data coverage in the tree list generation database: tgdb1r00

This section presents tables of summary statistics for the data coverage in the tree list generation database tgdb1r00 for untreated and thinned stands separated down by stand type. The statistics in the summary tables are derived from the scatterplot files generated by TGSCATRP for the tree list generation database tgdb1r00 and its untreated and thinned stand treatments.

The information in the summary tables describes the multidimensional histogram

bin centers defining the stand classification in the tree list generation database, and may be thought of as defining the range of stand conditions represented in the tree list generation database. Each multidimensional histogram bin represents a canonical or typical stand with the attributes of its bin center.

The tables of summary statistics can only provide a guide to the data that are actually contained within the tree list generation database `tgdb1r00` due to the high dimensionality of the classification, six (6) nontreatment index parameters for untreated stands and eleven (11) nontreatment index parameters for thinned stands. The relatively large dimension of these index parameter sets causes a combinatorial explosion in terms of presenting any sort of detailed printed data summary. Summarizing the data coverage in a tree list generation database is further complicated by the fact that it will contain gaps, and there is also no convenient manner in which to briefly summarize this information.

The summary information presented in the tables are the mean, standard deviation, minimum, median, maximum, and count for each index parameter of `INTERVAL` type for each treatment, broken down by stand type. Table 2.4 presents the stand type summary for the untreated and thinned stands in the tree list generation database `tgdb1r00`. The table summarizes the individual stand measurement files that comprise the tree list generation database, not the canonical stands represented by the multidimensional histogram bins, so the number of stands in this table need not agree with the canonical stand counts in the summary tables. A summary of the categorical index parameter `STAND_ORIGIN` does not appear in the general data summaries, but is provided by stand type in Table 2.5 and Table 2.6 for untreated and thinned stands, respectively. All of the data coverage summaries are in metric units. See Table 2.2 for the standard units for each type of data.

Table 2.4: Stand type summary based on the individual stand measurement files for the tree list generation database `tgdb1r00`. Percentages are rounded to the nearest 0.1%, so the row and column percent totals may not agree.

	Untreated stands		Thinned stands		Total	
Stand type	Count	Percent	Count	Percent	Count	Percent
PURE_DF	3404	35.3	3431	35.6	6835	70.8
PURE_WH	800	8.3	480	5.0	1280	13.3
DF_DOMINANT	691	7.2	417	4.3	1108	11.5
WH_DOMINANT	178	1.8	86	0.9	264	2.7
MIXTURE	136	1.4	26	0.3	162	1.7
Total	5209	54.0	4440	46.0	9649	100.0

Table 2.5: Stand origin summary for untreated stands, based on the individual stand measurement files for the tree list generation database `tgdb1r00`. Percentages are rounded to the nearest 0.1%, so the row and column percent totals may not agree.

	Natural		Planted		Total	
Stand type	Count	Percent	Count	Percent	Count	Percent
PURE_DF	2603	76.5	801	23.5	3404	65.3
PURE_WH	550	68.8	250	31.3	800	15.4
DF_DOMINANT	479	69.3	212	30.7	691	13.3
WH_DOMINANT	112	62.9	66	37.1	178	3.4
MIXTURE	118	86.8	18	13.2	136	2.6
Total	3862	74.1	1347	25.9	5209	100.0

Table 2.6: Stand origin summary for thinned stands, based on the individual stand measurement files for the tree list generation database `tgdb1r00`. Percentages are rounded to the nearest 0.1%, so the row and column percent totals may not agree.

	Natural		Planted		Total	
Stand type	Count	Percent	Count	Percent	Count	Percent
PURE_DF	2186	63.7	1245	36.3	3431	77.3
PURE_WH	449	93.5	31	6.5	480	10.8
DF_DOMINANT	372	89.2	45	10.8	417	9.4
WH_DOMINANT	86	100.0	0	0.0	86	1.9
MIXTURE	23	88.5	3	11.5	26	0.6
Total	3116	70.1	1324	29.8	4440	100.0

2.10.1 Data coverage summary for untreated stands

Table 2.7: Stand coverage summary for untreated stands and the stand type PURE_DF. There are 1921 canonical stands or multidimensional histogram bins in the stand classification for this stand type. Metric units are used for this summary.

Index parameter	Mean	SD	Minimum	Median	Maximum
SITE_INDEX_50	32.613	7.126	16.500	31.500	49.500
STAND_TOTAL_AGE	49.704	18.009	6.000	46.000	126.000
QMD	21.598	10.981	2.000	18.000	74.000
STAND_DENSITY	1650.963	1151.281	100.000	1300.000	9700.000

Table 2.8: Stand coverage summary for untreated stands and the stand type PURE_WH. There are 493 canonical stands or multidimensional histogram bins in the stand classification for this stand type. Metric units are used for this summary.

Index parameter	Mean	SD	Minimum	Median	Maximum
SITE_INDEX_50	36.946	9.898	19.500	34.500	58.500
STAND_TOTAL_AGE	46.876	16.252	18.000	46.000	94.000
QMD	17.578	7.870	6.000	14.000	42.000
STAND_DENSITY	3692.292	2852.759	700.000	2700.000	14100.000

Table 2.9: Stand coverage summary for untreated stands and the stand type DF_DOMINANT. There are 491 canonical stands or multidimensional histogram bins in the stand classification for this stand type. Metric units are used for this summary.

Index parameter	Mean	SD	Minimum	Median	Maximum
SITE_INDEX_50	33.577	6.723	16.500	34.500	43.500
STAND_TOTAL_AGE	48.998	16.554	14.000	46.000	106.000
QMD	18.774	8.034	6.000	18.000	50.000
STAND_DENSITY	2210.387	1397.065	100.000	1900.000	9500.000

Table 2.10: Stand coverage summary for untreated stands and the stand type WH_DOMINANT. There are 138 canonical stands or multidimensional histogram bins in the stand classification for this stand type. Metric units are used for this summary.

Index parameter	Mean	SD	Minimum	Median	Maximum
SITE_INDEX_50	39.609	8.019	25.500	37.500	55.500
STAND_TOTAL_AGE	48.435	15.118	18.000	46.000	86.000
QMD	21.275	6.595	10.000	22.000	42.000
STAND_DENSITY	2056.522	1107.569	700.000	1700.000	5500.000

Table 2.11: Stand coverage summary for untreated stands and the stand type MIXTURE. There are 96 canonical stands or multidimensional histogram bins in the stand classification for this stand type. Metric units are used for this summary.

Index parameter	Mean	SD	Minimum	Median	Maximum
SITE_INDEX_50	38.344	6.011	28.500	40.500	46.500
STAND_TOTAL_AGE	53.750	11.920	22.000	54.000	74.000
QMD	22.333	6.583	10.000	22.000	38.000
STAND_DENSITY	1535.417	1182.680	300.000	1000.000	4700.000

2.10.2 Data coverage summary for thinned stands

Table 2.12: Stand coverage summary for thinned stands and the stand type PURE_DF. There are 2820 canonical stands or multidimensional histogram bins in the stand classification for this stand type. Metric units are used for this summary.

Index parameter	Mean	SD	Minimum	Median	Maximum
SITE_INDEX_50	35.769	6.435	19.500	37.500	49.500
STAND_TOTAL_AGE	55.079	16.545	22.000	54.000	126.000
NUMBER_OF_THINNINGS	2.640	1.721	1.000	2.000	7.000
PRE_THIN_DENSITY	1139.149	1130.681	100.000	700.000	6100.000
PRE_THIN_BASAL_AREA	56.938	24.407	2.250	56.250	167.250
PCT_BASAL_AREA_REMOVED	32.414	13.805	0.750	32.250	95.250
YEARS_SINCE_TREATMENT	5.645	6.052	2.000	2.000	58.000
QMD	29.169	13.781	2.000	26.000	82.000
STAND_DENSITY	622.837	482.459	100.000	500.000	4100.000

Table 2.13: Stand coverage summary for thinned stands and the stand type PURE_WH. There are 424 canonical stands or multidimensional histogram bins in the stand classification for this stand type. Metric units are used for this summary.

Index parameter	Mean	SD	Minimum	Median	Maximum
SITE_INDEX_50	31.804	2.956	25.500	31.500	40.500
STAND_TOTAL_AGE	55.972	14.735	26.000	54.000	94.000
NUMBER_OF_THINNINGS	1.384	0.878	1.000	1.000	4.000
PRE_THIN_DENSITY	2983.019	2475.658	300.000	2100.000	12700.000
PRE_THIN_BASAL_AREA	53.597	18.478	5.250	57.750	81.750
PCT_BASAL_AREA_REMOVED	36.651	16.986	5.000	35.000	85.000
YEARS_SINCE_TREATMENT	4.519	3.340	2.000	2.000	14.000
QMD	22.811	8.100	6.000	22.000	50.000
STAND_DENSITY	1187.736	785.230	300.000	900.000	3900.000

2.11 File types, file formats, and keyword definitions

All of the tree list generation database input and output files are plain text files. The tree list generation database input and output file formats have been designed to be

Table 2.14: Stand coverage summary for thinned stands and the stand type DF_DOMINANT. There are 370 canonical stands or multidimensional histogram bins in the stand classification for this stand type. Metric units are used for this summary.

Index parameter	Mean	SD	Minimum	Median	Maximum
SITE_INDEX_50	36.219	5.575	19.500	37.500	40.500
STAND_TOTAL_AGE	53.276	14.509	26.000	50.000	94.000
NUMBER_OF_THINNINGS	2.608	1.891	1.000	2.000	6.000
PRE_THIN_DENSITY	1454.595	1248.041	100.000	1000.000	5900.000
PRE_THIN_BASAL_AREA	35.518	18.373	3.750	32.250	95.250
PCT_BASAL_AREA_REMOVED	24.757	16.518	5.000	25.000	95.000
YEARS_SINCE_TREATMENT	5.114	4.599	2.000	2.000	34.000
QMD	25.459	10.798	6.000	22.000	54.000
STAND_DENSITY	812.973	508.885	100.000	700.000	2700.000

Table 2.15: Stand coverage summary for thinned stands and the stand type WH_DOMINANT. There are 77 canonical stands or multidimensional histogram bins in the stand classification for this stand type. Metric units are used for this summary.

Index parameter	Mean	SD	Minimum	Median	Maximum
SITE_INDEX_50	36.019	2.315	31.500	37.500	37.500
STAND_TOTAL_AGE	55.766	17.466	34.000	50.000	94.000
NUMBER_OF_THINNINGS	2.675	1.902	1.000	2.000	6.000
PRE_THIN_DENSITY	1048.052	772.148	300.000	900.000	3500.000
PRE_THIN_BASAL_AREA	41.036	25.058	17.250	30.750	84.750
PCT_BASAL_AREA_REMOVED	22.013	15.223	5.000	15.000	45.000
YEARS_SINCE_TREATMENT	4.286	3.752	2.000	2.000	14.000
QMD	28.338	10.164	10.000	26.000	50.000
STAND_DENSITY	575.325	253.479	300.000	500.000	1100.000

Table 2.16: Stand coverage summary for thinned stands and the stand type MIXTURE. There are 25 canonical stands or multidimensional histogram bins in the stand classification for this stand type. Metric units are used for this summary.

Index parameter	Mean	SD	Minimum	Median	Maximum
SITE_INDEX_50	36.780	3.900	28.500	37.500	40.500
STAND_TOTAL_AGE	55.600	14.832	34.000	50.000	86.000
NUMBER_OF_THINNINGS	2.640	1.977	1.000	2.000	6.000
PRE_THIN_DENSITY	940.000	336.650	500.000	900.000	1500.000
PRE_THIN_BASAL_AREA	43.350	26.400	17.250	33.750	90.750
PCT_BASAL_AREA_REMOVED	12.600	9.256	5.000	15.000	45.000
YEARS_SINCE_TREATMENT	3.440	3.029	2.000	2.000	10.000
QMD	26.960	7.684	14.000	26.000	38.000
STAND_DENSITY	684.000	190.788	300.000	700.000	1100.000

easily understood by human readers as well as easily parsed by software. The file formats are line oriented, meaning that the fundamental unit in a file is a line of text. The files also generally follow a *keyword equals value* format, where a keyword and its value must appear on the same line separated by an equal sign (=). The keywords may appear in any order within a particular tree list generation database file. Blank lines and comment lines, indicated by a percent symbol (%) as the first nonblank character of a line, are ignored.

The file formats are straightforward being easy to modify and understand. The ability to place comments into the files without affecting the data provides a mechanism for annotating files used for a particular task. This feature, if used, could provide invaluable information about particular data sets at a later time. The data files also provide an exact record of the input and output for a particular analysis, and may be placed in an archive, allowing the reconstruction of the analysis at a later time, if this were to become necessary.

2.11.1 Stand measurement file

The tree list generation database *stand measurement file*, or measurement file, is a text file containing a description of a forest stand and an actual tree list. The tree list consists of DBH and height measurements with an indication of species for each tree in a measurement data set obtained from a sample plot. The stand measurement file uses a keyword equals value format and consists of two parts: a set of stand description keywords and values and the actual measured tree list data. The stand description keywords and their associated values describe the physical attributes of a measured stand and basic sample plot characteristics, e.g., site index, stand age, stand origin, sample plot size, treatment history, and the measurement units of the data. The measured tree list data are comprised of a DBH measurement, a height measurement, and a short two character species identification code, e.g., `DF`, or a longer four to six character species identification code, e.g., `PSME`, for each measured tree. All three items are required for each tree. Either the short or long species identification codes may be used in a stand measurement file, or they may be freely intermixed within a stand measurement file.

Each stand measurement file must contain a complete set of the stand description keywords for its treatment type, and each keyword must have an appropriate value. The stand description keywords may appear in any order within a stand measurement file. The stand description keywords are not case sensitive, so the keyword `TREATMENT_TYPE` is equivalent to the keyword `Treatment_Type`. Each stand description keyword must appear with its value on a single line in a stand measurement file.

The actual tree list data are the only exception to the keyword equals value format. The beginning of the tree list measurement data is indicated by the marker `BEGIN_TREE_DATA`, and the end of the tree list measurement data is indicated by the marker `END_TREE_DATA`. As with the stand description keywords , the case of the begin

and end markers is not important.

The individual tree data consists of DBH and height measurements and a species identifier, and the order of these data on the lines in the tree list data section of a stand measurement file must be specified. The first nonblank, noncomment line following the `BEGIN_TREE_DATA` marker must contain the tree list data column definitions indicating the order of the tree measurement data columns. The tree list data column definitions are specified by some blank delimited combination of the three identifiers `DBH`, `HEIGHT`, and `SPECIES`, which specify the order of the DBH, height, and species data for each tree in the tree list.

The `END_TREE_DATA` marker must be present; there is no implied end to the tree list data when a new keyword or the end of file is encountered. Once the actual tree list data have been identified by the `BEGIN_TREE_DATA` marker, only tree list data, one set of blank delimited DBH, height, and species values per line for each tree, in the order specified by the data column definitions, may appear, until the `END_TREE_DATA` marker is encountered. See Figure 2.30 and Figure 2.31 for stand measurement file templates for untreated and thinned stands, respectively.

In the stand measurement file templates for untreated and thinned stands, Figure 2.30 and Figure 2.31, respectively, a vertical bar | indicates that one of the two or more possible values is to be selected, and a numeric value must replace the marker `<number>`. The N sets of tree DBH, height and species markers must be replaced by appropriate numeric DBH and height measurements and species codes. The individual tree DBH and height measurement data values in a stand measurement file *must* be appropriate for the measurment units defined for the file.

Each stand measurement file must contain a `MEASUREMENT_UNITS` keyword. This keyword indicates the units of measure for the numerical data contained within a stand measurement file. The measurement units keyword may have a value of `IMPERIAL` or `METRIC`, depending on the measurement system used to represent the data. See Table 2.2 for the standard measurement units for each measured item. The tree

```
MEASUREMENT_UNITS       = IMPERIAL | METRIC
TREATMENT_TYPE          = THINNED
SITE_INDEX_50           = <number>
STAND_TOTAL_AGE         = <number>
STAND_ORIGIN            = PLANTED | NATURAL
PLOT_SIZE               = <number>
BEGIN_TREE_DATA
    DBH   HEIGHT   SPECIES
   DBH_1 HEIGHT_1 SPECIES_1
   DBH_2 HEIGHT_2 SPECIES_2
   DBH_3 HEIGHT_3 SPECIES_3
              .
              .
              .
   DBH_N HEIGHT_N SPECIES_N
END_TREE_DATA
```

Figure 2.30: Stand measurement file template for untreated stands. A vertical bar | separates possible values and indicates that one of the two or more possible values is to be selected, and a numeric value must replace the marker `<number>`.

list generation database software will automatically perform any unit conversions necessary when reading a stand measurement file. Internally the tree list generation database uses metric units to represent all of the tree and stand measurement data.

All stand measurement files. regardless of treatment, must contain the following stand description keywords.

`TREATMENT_TYPE` A text valued keyword that defines the type of treatment, or treatments, applied to a stand. Valid treatment types are `UNTREATED` and `THINNED`.

`SITE_INDEX_50` A numeric valued keyword for the site index of Douglas-fir, western hemlock, or mixed Douglas-fir and western hemlock stands at a reference stand age of 50 years. If a stand is not pure, i.e., the basal area for a dominant species is $< 80\%$, the site index value is for the species with the largeest percentage of basal area. The value for this keyword must be greater than zero; no other verification of the value is performed. The value specified should be appropriate for the measurement units.

```
MEASUREMENT_UNITS       = IMPERIAL | METRIC
TREATMENT_TYPE          = THINNED
SITE_INDEX_50           = <number>
STAND_TOTAL_AGE         = <number>
STAND_ORIGIN            = PLANTED | NATURAL
NUMBER_OF_THINNINGS     = <number>
PRE_THIN_DENSITY        = <number>
PRE_THIN_BASAL_AREA     = <number>
PCT_STEMS_REMOVED       = <number>
PCT_BASAL_AREA_REMOVED  = <number>
YEARS_SINCE_TREATMENT   = <number>
PLOT_SIZE               = <number>
BEGIN_TREE_DATA
    DBH   HEIGHT   SPECIES
   DBH_1 HEIGHT_1 SPECIES_1
   DBH_2 HEIGHT_2 SPECIES_2
   DBH_3 HEIGHT_3 SPECIES_3
              .
              .
              .
   DBH_N HEIGHT_N SPECIES_N
END_TREE_DATA
```

Figure 2.31: Stand measurement file template for thinned stands. A vertical bar | separates possible values and indicates that one of the two or more possible values is to be selected, and a numeric value must replace the marker `<number>`.

`STAND_AGE` A numeric valued keyword for the age in years of the stand of trees. This value must be greater than zero.

`STAND_ORIGIN` A text valued keyword that defines the origin of a stand. Valid stand origins are `NATURAL` and `PLANTED`. If a stand was seeded it is assumed to have a natural origin.

`PLOT_SIZE` A numeric valued keyword that specifies the sample plot size for the tree measurement data in the tree list. This is necessary to permit the scaling of certain values to a per unit area, hectare or acre, basis. The value specified should be appropriate for the measurement units. The value for this keyword must be greater than zero and less than 10.

Stand measurement files for treated stands must also contain stand description keywords which define the treatment history of the stand. The following stand description keywords describe the thinning history of a stand and are required for measurement files from thinned stands.

`NUMBER_OF_THINNINGS` A numeric valued keyword that defines the number of thinnings that have been performed on a stand. This value must be greater than zero, otherwise no thinnings have been performed.

`PRE_THIN_DENSITY` A numeric valued keyword defining the stand density as trees per unit area, acre or hectare, depending on the measurement units before the thinning. The value specified should be appropriate for the measurement units. The value of this keyword must be greater than zero. No other input consistency checks are performed at this time.

`PRE_THIN_BASAL_AREA` A numeric valued keyword defining the stand basal area per unit area, hectare or acre, depending on the measurement units before the

thinning. The value specified should be appropriate for the measurement units. The value of this keyword must be greater than zero. No other input consistency checks are performed at this time.

`PCT_STEMS_REMOVED` A numeric valued keyword defining the percentage of stems removed during the thinning. The value of this keyword must be greater than zero and less than 100. The upper bound of 100 is not currently enforced by the tree list generation database software. No other input consistency checks are performed at this time.

`PCT_BASAL_AREA_REMOVED` A numeric valued keyword defining the percentage of basal area removed during the thinning. The value of this keyword must be greater than zero and less than 100. The upper bound of 100 is not currently enforced by the tree list generation database software. No other input consistency checks are performed at this time.

`YEARS_SINCE_TREATMENT` A numeric valued keyword defining the number of years since the last thinning. The value of this keyword must be greater than zero.

2.11.2 Stand description file

The tree list generation database *stand description file*, or description file, is a text file defining the attributes of a forest stand that is to be generated using a tree list generation database. The stand description file follows a strict keyword equals value format using the *stand index parameter* keywords and their associated values to define the gross, aggregate characteristics of a forest stand. Each index parameter keyword must appear on a line with an appropriate value, delimited by an equal sign (=). In addition to the index parameter keywords, there are several optional keywords, specific to the stand description file, which provide additional information to the program `TGRAND` when it is used to generate a simulated stand or tree list. As is true

```
TREATMENT_TYPE
SITE_INDEX_50
STAND_TOTAL_AGE
STAND_ORIGIN
QMD
STAND_DENSITY
STAND_TYPE
```

Figure 2.32: Stand description file index parameters required for the `UNTREATED` treatment type in the tree list generation database `tgdb1r00`.

with all of the tree list generation database input files, blank lines and comment lines, indicated by a line whose first nonblank character is a percent sign (%), are ignored when the files are processed.

Each stand description file must contain a complete set of index parameter keywords for its treatment type and the tree list generation database in use when generating simulated stands or tree lists. Each index parameter keyword must also have an appropriate value. The stand description keywords may appear in any order within a stand description file. The stand description keywords are not case sensitive, so the keyword `TREATMENT_TYPE` is equivalent to the keyword `Treatment_Type`. Each stand description keyword must appear with its value on a single line in a stand description file. For the tree list generation database `tgdb1r00`, the required index parameters are listed in Figure 2.32 and Figure 2.33, for untreated and thinned stands, respectively. Index parameters keywords not used in a particular tree list generation database may also appear in a stand description file, providing a more complete stand description, possibly for later use. In this situation the *extra* index parameter keywords are ignored.

Figure 2.34 and Figure 2.35 provide examples of stand description files for untreated and thinned stands, respectively. In these two examples, the stand description files contain all of the index parameter keywords, even though only those listed in Figure 2.32 and Figure 2.33 were required for each treatment. The complete set

```
TREATMENT_TYPE
SITE_INDEX_50
STAND_TOTAL_AGE
STAND_ORIGIN
QMD
STAND_DENSITY
STAND_TYPE
NUMBER_OF_THINNINGS
PRE_THIN_DENSITY
PRE_THIN_BASAL_AREA
PCT_BASAL_AREA_REMOVED
YEARS_SINCE_TREATMENT
```

Figure 2.33: Stand description file index parameters required for the `THINNED` treatment type in the tree list generation database `tgdb1r00`.

of optional keywords is also used in these examples. The two stand description files presented in the figures were used to generate simulated stands for the testing and validation of the tree list generation database system using the tree list generation database `tgdb1r00`. The keywords beginning with "`RANDOM_MF_`" are the optional keywords, and provide information defining the processing modes of `TGRAND` and other items that affect the process of generating a simulated stand or tree list.

The `MEASUREMENT_UNITS` keyword must be contained in each tree list generation database stand description file. This keyword indicates the units of measure for the numerical data contained within a stand description file. The measurement units keyword may have a value of `IMPERIAL` or `METRIC`, depending on the measurement system used to represent the data. See Table 2.2 for the standard measurement units for each measured item. The tree list generation database software will automatically perform any unit conversions necessary when reading a schema file. Internally the tree list generation database uses metric units to represent all of the tree and stand measurement data.

The optional keywords for a stand description file are defined below. All of the option keywords begin with the string "`RANDOM_MF_`" to indicate that they are used

```
%
%  Tree list generator stand description file.
%
%  File created: 11/03/1999 at 16:22:54.52
%
MEASUREMENT_UNITS             = IMPERIAL
TREATMENT_TYPE                = UNTREATED
SITE_INDEX_50                 =  134.100
STAND_TOTAL_AGE               =   60.000
STAND_ORIGIN                  = NATURAL
MEAN_DBH                      =   10.835
MEAN_HEIGHT                   =   82.765
QMD                           =   12.442
TOP_HEIGHT                    =  124.125
STAND_DENSITY                 =  240.000
STAND_BASAL_AREA              =  202.627
PCT_DF_STEMS                  =   62.500
PCT_WH_STEMS                  =    4.167
PCT_OT_STEMS                  =   33.333
PCT_DF_BASAL_AREA             =   93.962
PCT_WH_BASAL_AREA             =    0.233
PCT_OT_BASAL_AREA             =    5.805
STAND_TYPE                    = PURE_DF
NUMBER_OF_SPECIES             =    3.000
RANDOM_MF_UNITS               = IMPERIAL
RANDOM_MF_PLOT_SIZE           =    0.200
RANDOM_MF_PURE                = NO
RANDOM_MF_JITTER              = NO
RANDOM_MF_NAME                = ptb00001.rmf
```

Figure 2.34: Example stand description file for untreated stands. This stand description file contains all of the possible index parameters with their values for the `UNTREATED` treatment type. Index parameters not used in a particular tree list generation database but appearing in a description file are ignored.

```
%
%  Tree list generator stand description file.
%
%  File created: 11/03/1999 at 16:21:09.60
%
MEASUREMENT_UNITS              = IMPERIAL
TREATMENT_TYPE                 = THINNED
SITE_INDEX_50                  =  134.100
STAND_TOTAL_AGE                =   60.000
STAND_ORIGIN                   = NATURAL
NUMBER_OF_THINNINGS            =    1.000
PRE_THIN_DENSITY               =  160.000
PRE_THIN_BASAL_AREA            =  178.900
PCT_STEMS_REMOVED              =   18.750
PCT_BASAL_AREA_REMOVED         =   14.925
YEARS_SINCE_TREATMENT          =    0.000
MEAN_DBH                       =   13.954
MEAN_HEIGHT                    =  102.292
QMD                            =   14.653
TOP_HEIGHT                     =  118.650
STAND_DENSITY                  =  130.000
STAND_BASAL_AREA               =  152.233
PCT_DF_STEMS                   =   92.308
PCT_WH_STEMS                   =    0.000
PCT_OT_STEMS                   =    7.692
PCT_DF_BASAL_AREA              =   98.744
PCT_WH_BASAL_AREA              =    0.000
PCT_OT_BASAL_AREA              =    1.256
STAND_TYPE                     = PURE_DF
NUMBER_OF_SPECIES              =    2.000
RANDOM_MF_UNITS                = IMPERIAL
RANDOM_MF_PLOT_SIZE            =    0.200
RANDOM_MF_PURE                 = NO
RANDOM_MF_JITTER               = NO
RANDOM_MF_NAME                 = ptb00001.rmf
```

Figure 2.35: Example stand description file for thinned stands. This stand description file contains all of the possible index parameters with their values for the `THINNED` treatment type. Index parameters not used in a particular tree list generation database but appearing in a description file are ignored.

to affect the output random stand measurement file that is produced by TGRAND. The complete set of index parameter keywords, other than the description file specific keywords presented here, are defined in Section 2.11.6.

RANDOM_MF_NAME A text valued keyword that defines the name of the output random stand measurement file used by TGRAND to store the simulated tree list. The file name specified *must* be the name of a new file. If a path is specified as a part of the file name, all of the path elements must exist. If this keyword is not defined, a default output filename of output.rmf is used, provided TGRAND is being used in its single file processing mode, otherwise it is an error to not define this keyword.

RANDOM_MF_UNITS A text valued keyword that defines the output units to be used in the random stand measurement file created by TGRAND. Valid values for this keyword are IMPERIAL or METRIC; any other value will result in an error. If this keyword is not defined, the measurement units in the random measurement file will be the same as the measurement units in the stand description file used.

RANDOM_MF_PURE A text valued keyword that defines whether 100% pure stands are to be generated for pure Douglas-fir or pure western hemlock stands. Valid values for this keyword are YES or NO; any other value will result in an error. 100% pure stands of Douglas-fir will be generated if this keyword has a value of YES *and* the stand type is PURE_DF, and 100% pure stands of western hemlock will be generated if this keyword has a value of YES *and* the stand type is PURE_WH. Stands with a stand type other than PURE_DF and PURE_WH are not affected by the value of this keyword. If this keyword is not defined, the default value is to not generate 100% pure stands. See Section 3.1.4 for the procedure used to generate 100% pure stands.

`RANDOM_MF_JITTER` A text valued keyword that defines whether individual tree heights are to be jittered. Valid values for this keyword are `YES` or `NO`; any other value will result in an error. The jittering of tree heights could provide a better tree height distribution, since most of the individual tree heights stored in a tree list generation database will be estimated from height-diameter curves. If this keyword is not defined, the default is to not jitter heights. See Section 3.1.4 for the procedure used to generate trees with jittered heights.

`RANDOM_MF_PLOT_SIZE` A numeric valued keyword that specifies the output plot size for the simulated tree list. This is used to generate the appropriate number of trees per unit area, hectare or acre, for the simulated stand. The value specified should be appropriate for stand description file measurement units, *not* the units specified by the `RANDOM_MF_UNITS` keyword. The value for this keyword must be greater than zero and less than 10. If this keyword is not defined, the default value used is 0.1 unit, hectare or acre.

The `RANDOM_MF_NAME` keyword is required if `TGRAND` is being used in its *batch processing mode* with a *listing file*, indicated by the `-lf` command line option. If this keyword is not defined and `TGRAND` is being used in batch mode, an error will occur and batch processing will halt. If the `RANDOM_MF_NAME` keyword is not defined and `TGRAND` is being used in its *single file mode*, indicated by the `-df` command line option, a filename of `output.rmf` is used by default.

If the `RANDOM_MF_PLOT_SIZE` keyword is used to define the size of the area represented by the simulated tree list, it is critical that its numeric value be in the appropriate units. The units for *all* of the numeric valued keywords in a description file are defined by the value of the `MEASUREMENT_UNITS` keyword for the description file, *not* by the units specified by the `RANDOM_MF_UNITS` keyword. The `RANDOM_MF_UNITS` keyword is used to specify the measurement units for the output random measurement file, and may be used to automatically perform a units conversion if the measure-

ment units of the description file are different than the measurement units desired for the random measurement file. As an example, suppose that the description file has a value of METRIC for the MEASUREMENT_UNITS keyword, and a value of IMPERIAL for the RANDOM_MF_UNITS keyword. A value of 0.1 (ha) for the RANDOM_MF_PLOT_SIZE keyword would produce an output plot size of 0.247 (ac).

When a stand description file is used to generate a simulated stand using TGRAND, values of some of the index parameter keywords in the description file are simply copied, possibly with a change of units, to the random stand measurement file that is created. The index parameter keyword values that are copied for all treatment types are:

TREATMENT_TYPE
SITE_INDEX_50
STAND_TOTAL_AGE
STAND_ORIGIN,

and the for the thinned treatment type, the additional index parameter keyword values that are copied to the random measurement file are:

NUMBER_OF_THINNINGS
PRE_THIN_DENSITY
PRE_THIN_BASAL_AREA
PCT_STEMS_REMOVED
PCT_BASAL_AREA_REMOVED
YEARS_SINCE_TREATMENT.

These index parameter values are copied from a stand description file to a random stand measurement file for two reasons. First, the values specified in the stand description file are the known or desired values for those particular stand attributes in the simulated stand, so copying them makes sense. Second, the multidimensional histogram based stand classification procedure combines data from similar stands and refers to them by the center of the histogram bin. Thus exact values for these index parameters are not easily available. Further, having exact values for these index parameters would not necessarily be particularly useful in this context. For example, what stand age should be given to a simulated stand if the exact stand ages were

available for all of the stands in a multidimensional histogram bin? The average age of all stands? The closest age to the desired age? The simplest, and most appropriate, resolution of this issue is to simply copy the index parameter values from the stand description file to the output random stand measurement file, as is done, using the desired stand attribute values.

2.11.3 Random stand measurement file

The tree list generation database *random stand measurement file*, or random measurement file, is a text file that contains a set of stand description keywords and a simulated tree list obtained from a tree list generation database. The format of the random measurement file is identical to that of the stand measurement file, Section 2.11.1; both file types contain the same kind of data, so the same file format is used. A random stand measurement file obtained from a stand description file, Section 2.11.2, using the tree list generation database program `TGRAND`, Section 2.4, will contain a comment indicating that it is a random stand measurement file, see Figure 2.36. At this time this is the only method for distinguishing the contents of a measurement file as being simulated data. Thus, some care must be taken to not confuse simulated data in random measurement files with actual stand measurement data.

When a random stand measurement file is created using `TGRAND` and a stand description file, values of some of the index parameter keywords in the description file are simply copied, possibly with a change of units, to the random stand measurement file. The index parameter keyword values that are copied for all treatment types are:

```
TREATMENT_TYPE
SITE_INDEX_50
STAND_TOTAL_AGE
STAND_ORIGIN,
```

and the for the thinned treatment type, the additional index parameter keyword values that are copied to the random measurement file are:

```
%
%  Tree list generator random stand measurement file.
%
%  File created: 12/07/1999 at 12:24:11.63
%
MEASUREMENT_UNITS              = METRIC
TREATMENT_TYPE                 = UNTREATED
SITE_INDEX_50                  = 24.07900
STAND_TOTAL_AGE                = 54.00000
STAND_ORIGIN                   = NATURAL
PLOT_SIZE                      =  0.06100
BEGIN_TREE_DATA
     DBH   HEIGHT  SPECIES
   44.291  27.373    DF
   27.908  23.847    DF
        .
        .
        .
END_TREE_DATA
```

Figure 2.36: Random stand measurement file example with the file header comments. These comments are currently the only way of distinguishing a random measurement file containing simulated data from a stand measurement file containing actual data, so some care must be taken that files containing simulated data do not become confused with files containing actual data.

NUMBER_OF_THINNINGS
PRE_THIN_DENSITY
PRE_THIN_BASAL_AREA
PCT_STEMS_REMOVED
PCT_BASAL_AREA_REMOVED
YEARS_SINCE_TREATMENT.

See Section 2.11.2 for an explanation of why these values are simply copied from the description file to the random measurement file.

2.11.4 Species mapping file

The tree list generation database *species mapping file* defines an association between unique alphabbetic,or character based, codes for the identification of tree species and an unique integer identification code for each tree species that is recognized by the tree list generation database system. The alphabetic tree species codes are two (2) or four (4) to six (6) character codes in common use for identifying tree species, and the integer identification codes are used internally by the tree list generation database software. In addition to the species identification codes, the species mapping file also contains the common and Latin names for each recognized tree species. The tree list generation database species mapping file is used when creating a tree list generation database, and it defines the tree species that will be recognizable for a particular tree list generation database. Tree species codes that are not contained in the species mapping file used to create a tree list generation database will not be recognized, and will cause an error to be signalled by the programs requiring the species information for their use.

The tree list generation database species mapping file should *not* be modified by any users of the tree list generation database executable programs. Additions or modifications made to the species mapping file must be done with extreme care. If modifications to this file are made incorrectly, its use with the tree list generation database software may produce unpredictable results. In particular, it is important that all integer and character based species codes are unique. Two situations where

modifications to the tree list generation database species mapping file are warranted will be discussed shortly.

The tree list generation database species mapping file contains five comma delimited columns:

1. The unique integer tree species ID code.

2. The unique two (2) letter tree species code.

3. The unique four (4) to six (6) letter tree species code.

4. The common name for the species.

5. The Latin name for the species.

The commas in this file *are necessary*. Comment lines and blank lines are ignored. Comment lines are lines whose first nonblank character is a percent sign (%). Only the first three columns are currently used by the tree list generation database software. The common and Latin species names have been included for completeness. See Table 2.17 for the species codes and common names. The Latin names may be found in Appendix C.

There were several tree species code conflicts in the data, i.e., two different codes being used to represent the same tree species. To resolve these conflicts the species codes used by the SMC were selected for use in the tree list generation database.

- The tree species code `OH` for other hardwood was changed to be `HD`, the code used by the SMC. Further use of the tree species code `OH` is not recommended.

- The tree species code `PM` for pacific madrone *Arbutus menziesii* was changed to be `MD`, the code used by the SMC. Further use of the tree species code `PM` is not recommended.

- The tree species code `03` appeared in some of the data. A description of this species code could not be found, so it was changed to `OT`, the code used by the SMC for unrecognized or other species. Further use of the tree species code `03` is not recommended.

These decisions were made because it is in the best, long term interests of the SMC cooperators to standardize the various tree species codes used, making the transfer of data among the cooperators easier and more reliable. If the data used to create the tree list generation database are indicative of the data held by the SMC cooperators, there is greater than 99% agreement already, so these recommendations should have a minimal impact.

The two situations when modifying a tree list generation database species mapping file are warranted occur when a new tree species must be recognized by the tree list generation database software, and when the integer species codes must be changed to agree with the integer tree species codes used in other software. In the first situation, a new set of species codes may be added to the end of the file by incrementing the last integer species code by one (1) and completing the rest of the comma delimited fields for the definition of the new species. When adding a new tree species, the two (2) and four (4) to six (6) character species codes added *must* be unique. In the second situation, the integer species codes in the file may be changed provided that the integer codes added are unique. In both instances, a *new* tree list generation database must be created using the modified species mapping file. The use of the modified integer ID codes with an existing tree list generation database will cause incorrect identification of the tree species codes.

The executable program `TGXPLODE` may be used first to break an existing tree list generation database into its component files. The executable programs `TGNEW` and `TGADD` may then be used with a new species mapping file and the component files extracted by `TGXPLODE` to build a new tree list generation database that will recognize

all of the species.

2.11.5 Schema file

The tree list generation database *schema file* is a text file defining the structure of a tree list generation database that is to be created. The tree list generation database schema file is analogous to a database schema that is used to define the number and types of data columns in a database. The tree list generation database schema file defines which treatment types and stand classification variables, or *stand index parameters*, are used to create a particular tree list generation database. The schema file also defines the attributes of each stand index parameter for a particular treatment type: the index parameter type and the histogram bin width that is used for the stand classification and indexing. Index parameters may have one of two types: `FIXED` for stand attributes that are represented by categorical variables, such as stand origin or stand type, or `INTERVAL` for stand attributes that are represented by continuous variables, such as age, mean DBH, top height, or basal area per acre. The histogram bin width attribute is only used for index parameters which represent continuous variables and have an index parameter type of `INTERVAL`.

Each tree list generation database schema file is composed of three nested levels, each with its own scope. The first level is the schema file itself, which defines the treatment types that will be contained in a particular tree list generation database. The second level is the treatment definition level, which defines the index parameters that are to be used for a particular treatment in a particular tree list generation database. The third level is the index parameter definition level, and this is the level at which the attributes of each index parameter are defined for a treatment in a tree list generation database. The index parameter attributes indicate whether a particular index parameter is to be used for a treatment, the type of the index parameter, e.g., `FIXED` or `INTERVAL`, and the histogram bin width for any index parameter with an `INTERVAL` type.

Table 2.17: Tree list generation database species mapping table. The table presents the internal integer ID codes used by the software, a short two (2) character tree species code, a long four (4) to six (6) character tree species code, and the common name for a tree species. See Table B.1 for the latin names of the tree species. See also Appendix C for an example of a species mapping file.

ID Code	Short Code	Long Code	Common Name
1	BC	POTR	Black Cottonwood
2	BR	BEPA	Birch
3	CA	RHPU	Cascara
4	CH	PREM	Bitter Cherry
5	DF	PSME	Douglas-fir
6	DM	ACGL	Douglas Maple
7	DW	CONU	Western Flowering Dogwood
8	ES	PIEN	Englemann Spruce
9	GC	CACH	Golden Chinkapin
10	GF	ABGR	Grand Fir
11	HD	HD	Other hardwoods
12	HW	CRDO	Hawthorn
13	HZ	COCO2	Hazelnut
14	IC	LIDE	Incense Cedar
15	KP	PIAT	Knobcone Pine
16	LP	PICO	Lodgepole Pine
17	MA	ACMA	Bigleaf Maple
18	MD	ARME	Madrone
19	MY	UMCA	Oregon Myrtle
20	NF	ABPR	Noble Fir
21	OA	FRLA	Oregon Ash
22	OO	QUGA	Garry Oak
23	OT	OT	Other species or mixtures
24	PC	CHLA	Port Orford Cedar
25	PP	PIPO	Ponderosa Pine
26	RA	ALRU	Red Alder
27	RC	THPL	Western Red Cedar
28	RF	ABMA	California Red Fir
29	SA	ALCR2	Sitka Alder
30	SB	AMELA	Servicberry
31	SF	ABAM	Pacific Silver Fir
32	SP	PILA	Sugar Pine
33	SS	PISI	Sitka Spruce
34	SX	SALIX	Willow

Table 2.17: (continued)

35	TO	LIDE2	Tan Oak
36	VM	ACCI	Vine Maple
37	WF	ABCO	White Fir
38	WH	TSHE	Western Hemlock
39	WL	LAOC	Western Larch
40	WM	MYCA	Waxmyrtle
41	WP	PIMO	Western White Pine
42	WY	TABR	Western Yew
43	YC	CHNO	Alaska Yellow Cedar
44	AF	ABLA	Alpine fir
45	BO	QUKE	Black Oak

The tree list generation database schema file uses a slightly modified keyword equals value format. Treatment type definitions and index parameter attribute definitions each have an end marker, `END_TREATMENT` or `END_INDEX_PARAMETER`, respectively, to denote the end of the particular nesting level or scope. The end of the a schema file denotes the end of the treatment type definition level. Index parameter attributes are defined in a strict keyword equals value manner, with the keyword and value both appearing on the same line in the file delimited by an equal sign (=). The treatment type definition level and index parameter definition level of a schema file begin with a keyword equals value notation and the keywords `TREATMENT_TYPE` and `INDEX_PARAMETER`, respectively. As is true with all of the tree list generation database input files, blank lines and comment lines, indicated by a line whose first nonblank character is a percent sign (%), are ignored when the files are read.

Templates for the three nested levels in a tree list generation database schema file are presented in Figure 2.37, Figure 2.38, and Figure 2.39. The figures present increasing levels of detail, or nesting level. In the figures, a vertical bar "|" separates possible values for an index parameter attribute keyword, and indicates that one, and only one of the two or more possible values is to be selected. The marker `<number>` should be replaced by an appropriate nonzero numerical value. The square brackets

```
MEASUREMENT_UNITS = IMPERIAL | METRIC

TREATMENT_TYPE = <treatment_1>
   <index parameter definitions for treatment_1>
END_TREATMENT

TREATMENT_TYPE = <treatment_1>
   <index parameter definitions for treatment_2>
END_TREATMENT
                      .
                      .
                      .
TREATMENT_TYPE = <treatment_n>
   <index parameter definitions for treatment_n>
END_TREATMENT
```

Figure 2.37: Schema file template for the treatment types to be used in a tree list generation database. See Figure 2.38 for a the template for the index parameter definition level within each treatment.

enclosing the INTERVAL_WIDTH parameter attribute, indicate that it isonly required if the PARAMETER_TYPE has a value of INTERVAL, and should not be present for an index parameter with a PARAMETER_TYPE value of FIXED. Specifying the INTERVAL_WIDTH parameter attribute for an index parameter with a FIXED type will produce an error. Section 2.11.6 provides complete details on the available index parameters for each treatment.

Each tree list generation database schema file must contain a MEASUREMENT_UNITS keyword. This keyword indicates the units of measure for the numerical data contained within a schema file. The measurement units keyword may have a value of IMPERIAL or METRIC, depending on the measurement system used to represent the data. See Table 2.2 for the standard measurement units for each measured item. The tree list generation database software will automatically perform any unit conversions necessary when reading a schema file. Internally the tree list generation database uses metric units to represent all of the tree and stand measurement data.

The index parameters used with a particular treatment in a tree list generation database are selected via the schema file by setting the value of the USE_PARAMETER

```
TREATMENT_TYPE = <treatment_i>
   INDEX_PARAMETER = <index_parameter_1>
      <attribute definitions for index_parameter_1>
   END_INDEX_PARAMETER

   INDEX_PARAMETER = <index_parameter_2>
      <attribute definitions for index_parameter_2>
   END_INDEX_PARAMETER
                        .
                        .
                        .
   INDEX_PARAMETER = <index_parameter_m>
      <attribute definitions for index_parameter_m>
   END_INDEX_PARAMETER
END_TREATMENT
```

Figure 2.38: Schema file template for the index parameter definition level within `<treatment_i>`. There are `m` index parameters defined for this treatment. See Figure 2.39 the template for the index parameter attribute definition level within each index parameter.

```
                    .
                    .
                    .
  TREATMENT_TYPE = <treatment_i>
                    .
                    .
                    .
     INDEX_PARAMETER = <index_parameter_j>
        USE_PARAMETER   = YES | NO
        PARAMETER_TYPE  =  FIXED | INTERVAL
        [INTERVAL_WIDTH = <number>]
     END_INDEX_PARAMETER
                    .
                    .
                    .
  END_TREATMENT
                    .
                    .
                    .
```

Figure 2.39: Schema file template for the index parameter attribute definition level for `<index_parameter_j>` within `<treatment_i>`. The `INTERVAL_WIDTH` parameter is only used if the index parameter type is `INTERVAL`, otherwise it is not present.

attribute. A value of YES indicating that the index parameter should be used and a value of NO indicating that the index parameter should not be used. It is not necessary to specify all of the possible index parameters for a treatment in a schema file, only the index parameters that will actually be used in a tree list generation database for a treatment need to be defined. In this case, all of the index parameters will have a value of YES for their USE_PARAMETER attributes. Thus, index parameters that are not used for a treatment need not be included in a schema file, making the file smaller.

The order of treatment type definitions in a tree list generation database schema file, if there are multiple treatments, is not important, but each treatment specification must be complete, ending with the END_TREATMENT marker before a new treatment may begin. Likewise, the order of index parameter definitions within a treatment is not important, but each index parameter specification must be complete, ending with the END_INDEX_PARAMETER marker before a new index parameter definition may begin. Finally, the index parameter attributes may appear in any order within an index parameter definition for a treatment. Each index parameter attribute must have an appropriate value, and if the PARAMETER_TYPE has a value of INTERVAL, the INTERVAL_WIDTH attribute must be defined and have a nonzero, positive value.

The STAND_TYPE index parameter is required for all treatments. This index parameter is used to distinguish among the different categories of stands, e.g., pure Douglas-fir, pure western hemlock, douglas-fir and western hemlock mixtures, and stand with arbitrary species composition. If this index parameter were not required, stands of different species would be indistinguishable from each other; only their numeric attributes would be used to distinguish them from one another. Thus, not requiring the STAND_TYPE index parameter would defeat the purpose of distinguishing among different stand types for generating treee lists. This index parameter should have a PARAMETER_TYPE of FIXED, and its USE_PARAMETER value must always be YES. If the STAND_TYPE index parameter is not present within a treatment definition, an error will be signalled when reading the schema file.

When defining the index parameter attributes some care must be taken to avoid defining a categorical classification variable, indicated by a PARAMETER_TYPE value of FIXED, as a continuous classification variable, indicated by a PARAMETER_TYPE value of INTERVAL, or vice versa. The consequences of defining a categorical classification variable as a continuous classification variable would be to consider a range of categorical variables as being equivalent when classifying stands, which would obviously be disastrous for an index parameter such as STAND_TYPE which is based on species composition. The consequences of defining a continuous classification variable as a categorical classification variable would be to be to consider each value as unique, creating a separate category for every value, which would become quite cumbersome for an index parameter such as QMD or top height. The tree list generation database software does not currently impose particular types on the index parameters, but common sense and standard interpretations of the index parameter should be a sufficient guide when determining index parameter types.

When creating a tree list generation database which contains multiple treatments, all treatments should contain a uniform subset of index parameters with the same interval widths for the INTERVAL index parameters. This enforces an internal consistency across the treatments. For example, if a tree list generation database contains data for untreated stands and thinned stands, the index parameters and interval widths used for the untreated stands should be used for the thinned stands as well, with one or more additional index parameters specific to the thinned treatment. If this is not done, the treatments in a tree list generation database will not be classified consistently across treatments. There may, however, be instances where this latter situation is desired, so the tree list generation database software does not impose any type of consistency constraint across treatment types, permitting this type of use.

See Section 2.11.6 for the index parameter keywords, their definitions, and recommended types and values. The schema file used to create the tree list generation database tgdb1r00 may be found in Appendix D. An annotated schema file template

may be found in Appendix E.

2.11.6 Index paramter keyword definitions

The index parameter keywords used to define a multidimensional histogram stand classification for a tree list generation database, and to specify a stand description for a simulated stand, are defined in this section. Each index parameter definition contains the recommended index parameter type, the index parameter interval width for index parameters of type `INTERVAL`, a description of the set of valid values for the index parameter, the treatments that the index parameter may be applied to, an indication of whether the index parameter is used in the tree list generation database `tgdb1r00`, and a brief description of the index parameter keyword. A value of "N/A" indicates that there is no applicable value for that component of an index parameter definition. A rationale for the selection of the interval widths used for the index parameters of `INTERVAL` type is provided in the description for each such index parameter definition.

`TREATMENT_TYPE`

Index parameter type: N/A

Index parameter interval width: N/A

Valid index parameter values: One of `THINNED` or `UNTREATED`

Applicable Treatments: All treatment types

Used in `tgdb1r00`: Yes

Description: The treatment type. This index parameter keyword must always be present in a schema file or stand description file.

`SITE_INDEX_50`

Index parameter type: `INTERVAL`

Index parameter interval width: 3.0 m (8.4 ft)

Valid index parameter values: Any numeric value ≥ 0

Applicable Treatments: All treatment types

Used in `tgdb1r00`: Yes

Description: The site index at 50 years reference age for Douglas-fir or western hemlock stands. The interval width of 3.0 m (8.4 ft) is used because it is approximately one third of the width of a site class for Douglas-fir [15, 4].

`STAND_TOTAL_AGE`

Index parameter type: `INTERVAL`

Index parameter interval width: 4 years

Valid index parameter values: Any numeric value ≥ 0

Applicable Treatments: All treatment types

Used in `tgdb1r00`: Yes

Description: The total age of a forest stand. The interval width of four (4) years is used to avoid possible aliasing problems caused by an interval width that is too small, given the remeasurement intervals which ranged from two (2) to eight (8) years. This interval width also takes into account the fact that there is some uncertainty in the stand total ages for much of the data that are available.

`STAND_ORIGIN`

Index parameter type: `FIXED`

Index parameter interval width: N/A

Valid index parameter values: One of `NATURAL` or `PLANTED`

Applicable Treatments: All treatment types

Used in `tgdb1r00`**:** Yes

Description: The origin of a forest stand. Two stand origin types are provided, `NATURAL` for seeded and naturally regenerated stands, and `PLANTED` for planted stands.

`MEAN_DBH`

Index parameter type: `INTERVAL`

Index parameter interval width: 4 cm (1.57 in)

Valid index parameter values: Any numeric value ≥ 0

Applicable Treatments: All treatment types

Used in `tgdb1r00`**:** No

Description: The mean DBH of a forest stand. This index parameter is given a value that represents the arithmetic mean diameter at breast height of all trees in a stand. The interval width of 4 cm (1.57 in) is used because it is a commonly used value for determining diameter classes.

`MEAN_HEIGHT`

Index parameter type: `INTERVAL`

Index parameter interval width: 3 m (8.4 ft)

Valid index parameter values: Any numeric value ≥ 0

Applicable Treatments: All treatment types

Used in `tgdb1r00`**:** No

Description: The mean tree height of a forest stand. This index parameter is given a value that represents the arithmetic mean tree height of all trees in a stand. The interval width of 3.0 m (8.4 ft) is used because it is approximately one third of the width of a site class for Douglas-fir [15, 4].

`QMD`

Index parameter type: `INTERVAL`

Index parameter interval width: 4 cm (1.57 in)

Valid index parameter values: Any numeric value ≥ 0

Applicable Treatments: All treatment types

Used in `tgdb1r00`: Yes

Description: The quadratic mean diameter, QMD, of a forest stand. This index parameter is given a value that represents the diameter of a tree which has the mean tree basal area over all the trees in a stand. The interval width of 4 cm (1.57 in) is used because it is a commonly used value for determining diameter classes.

`TOP_HEIGHT`

Index parameter type: `INTERVAL`

Index parameter interval width: 3 m (8.4 ft)

Valid index parameter values: Any numeric value ≥ 0

Applicable Treatments: All treatment types

Used in `tgdb1r00`: No

Description: The mean height of the 100 (40) largest diameter trees per hectare (acre) in a forest stand. The interval width of 3.0 m (8.4 ft) is used

because it is approximately one third of the width of a site class for Douglas-fir [15, 4].

`STAND_DENSITY`

Index parameter type: `INTERVAL`

Index parameter interval width: 200 tph (80 tpa)

Valid index parameter values: Any numeric value ≥ 0

Applicable Treatments: All treatment types

Used in `tgdb1r00`: Yes

Description: The stand density in trees per hectare (acre) in a forest stand. The interval width of 200 tph (80 tpa) is used because it is approximately ± 4 measured trees per plot with an average plot size of 0.0405 ha (0.10 ac) or ± 10 trees per plot with an average plot size of 0.10 ha (0.25 ac).

`STAND_BASAL_AREA`

Index parameter type: `INTERVAL`

Index parameter interval width: 1.5 m^2ha^{-1} (16.15 ft^2ac^{-1})

Valid index parameter values: Any numeric value ≥ 0

Applicable Treatments: All treatment types

Used in `tgdb1r00`: No

Description: The basal area in per hectare (acre) for a forest stand. The interval width of 1.5 m^2ha^{-1} (16.15 ft^2ac^{-1}) is used because it approximates the annual basal area increment.

`PCT_DF_STEMS`

Index parameter type: INTERVAL

Index parameter interval width: 20%

Valid index parameter values: Any numeric value between zero (0) and 100

Applicable Treatments: All treatment types

Used in tgdb1r00: No

Description: The percentage of the trees or stems per hectare (acre) that are Douglas-fir trees. The interval width value of 20% is used because this is consistent with the definition of a pure stand as having at least 80% of the stand be the dominant species. If this index parameter is used the index parameters PCT_WH_STEMS and PCT_OT_STEMS must also be used.

PCT_WH_STEMS

Index parameter type: INTERVAL

Index parameter interval width: 20%

Valid index parameter values: Any numeric value between zero (0) and 100

Applicable Treatments: All treatment types

Used in tgdb1r00: No

Description: The percentage of the trees or stems per hectare (acre) that are western hemlock trees. The interval width value of 20% is used because this is consistent with the definition of a pure stand as having at least 80% of the stand be the dominant species. If this index parameter is used the index parameters: PCT_DF_STEMS and PCT_OT_STEMS must also be used.

PCT_OT_STEMS

Index parameter type: INTERVAL

Index parameter interval width: 20%

Valid index parameter values: Any numeric value between zero (0) and 100

Applicable Treatments: All treatment types

Used in `tgdb1r00`: No

Description: The percentage of the trees or stems per hectare (acre) that are *not* Douglas-fir or western hemlock trees. The interval width value of 20% is used because this is consistent with the definition of a pure stand as having at least 80% of the stand be the dominant species. If this index parameter is used the index parameters `PCT_DF_STEMS` and `PCT_WH_STEMS` must also be used.

`PCT_DF_BASAL_AREA`

Index parameter type: `INTERVAL`

Index parameter interval width: 20%

Valid index parameter values: Any numeric value between zero (0) and 100

Applicable Treatments: All treatment types

Used in `tgdb1r00`: No

Description: The percentage of the basal area per hectare (acre) that is composed of Douglas-fir trees. The interval width value of 20% is used because this is consistent with the definition of a pure stand as having at least 80% of the stand basal area being the dominant species. If this index parameter is used the index parameters `PCT_WH_BASAL_AREA` and `PCT_OT_BASAL_AREA` must also be used.

`PCT_WH_BASAL_AREA`

Index parameter type: `INTERVAL`

Index parameter interval width: 20%

Valid index parameter values: Any numeric value between zero (0) and 100

Applicable Treatments: All treatment types

Used in `tgdb1r00`: No

Description: The percentage of the basal area per hectare (acre) that is composed of western hemlock trees. The interval width value of 20% is used because this is consistent with the definition of a pure stand as having at least 80% of the stand basal area being the dominant species. If this index parameter is used the index parameters `PCT_DF_BASAL_AREA` and `PCT_OT_BASAL_AREA` must also be used

`PCT_OT_BASAL_AREA`

Index parameter type: `INTERVAL`

Index parameter interval width: 20%

Valid index parameter values: Any numeric value between zero (0) and 100

Applicable Treatments: All treatment types

Used in `tgdb1r00`: No

Description: The percentage of the basal area per hectare (acre) that is *not* composed of Douglas-fir or western hemlock trees. The interval width value of 20% is used because this is consistent with the definition of a pure stand as having at least 80% of the stand basal area being the dominant species. If this index parameter is used the index parameters `PCT_DF_BASAL_AREA` and `PCT_WH_BASAL_AREA` must also be used

`STAND_TYPE`

Index parameter type: FIXED

Index parameter interval width: N/A

Valid index parameter values: One of PURE_DF, PURE_WH, DF_DOMINANT, WH_DOMINANT, or MIXTURE

Applicable Treatments: All treatment types

Used in tgdb1r00: Yes

Description: The type of the stand based on species composition. The value of PURE_DF indicates that a stand is at least 80% Douglas-fir, PURE_WH indicates that a stand is at least 80% western hemlock, DF_DOMINANT indicates that a stand is at least 50% Douglas-fir, WH_DOMINANT indicates that a stand is at least 50% western hemlock, and MIXTURE indicates that a stand does not have a Douglas-fir or western hemlock percentage above 50%, and is therefore an arbrtrary mixture. The index parameter is required.

NUMBER_OF_SPECIES

Index parameter type: INTERVAL

Index parameter interval width: 5

Valid index parameter values: Any numeric value ≥ 0

Applicable Treatments: All treatment types

Used in tgdb1r00: No

Description: The number of species in a stand. The interval width of five (5) is used because it is common to have ingrowth from several species other than the desired dominant species, and the type of species is not necessarily important.

NUMBER_OF_THINNINGS

Index parameter type: FIXED

Index parameter interval width: N/A

Valid index parameter values: Any integer numeric value ≥ 0

Applicable Treatments: THINNED treatments

Used in tgdb1r00: Yes

Description: The number of thinnings that have been performed on a stand of trees.

PRE_THIN_DENSITY

Index parameter type: INTERVAL

Index parameter interval width: 200 tph (80 tpa)

Valid index parameter values: Any numeric value ≥ 0

Applicable Treatments: THINNED treatments

Used in tgdb1r00: Yes

Description: The number of trees per hectare (acre) on a forest stand before a thinning operation was performed. The interval width of 200 tph (80 tpa) is used because it is approximately ± 4 measured trees per plot with an average plot size of 0.0405 ha (0.10 ac) or ± 10 trees per plot with an average plot size of 0.10 ha (0.25 ac).

PRE_THIN_BASAL_AREA

Index parameter type: INTERVAL

Index parameter interval width: 1.5 $m^2 ha^{-1}$ (16.15 $ft^2 ac^{-1}$)

Valid index parameter values: Any numeric value ≥ 0

Applicable Treatments: `THINNED` treatments

Used in `tgdb1r00`: Yes

Description: The basal area per hectare (acre) on a forest stand before a thinning operation is performed. The basal area in per hectare (acre) for a forest stand. The interval width of 1.5 m^2ha^{-1} (16.15 ft^2ac^{-1}) is used because it approximates the annual basal area increment.

`PCT_STEMS_REMOVED`

Index parameter type: `INTERVAL`

Index parameter interval width: 10%

Valid index parameter values: Any numeric value between zero (0) and 100

Applicable Treatments: `THINNED` treatments

Used in `tgdb1r00`: No

Description: The percentage of trees or stems removed from a forest stand during a thinning operation. The interval width of 10% is used because the percentage of trees removed during a particular thinning event is expected to be less than 50% of the standing trees, and 10% interval widths should provide a reasonable resolution.

`PCT_BASAL_AREA_REMOVED`

Index parameter type: `INTERVAL`

Index parameter interval width: 10%

Valid index parameter values: Any numeric value between zero (0) and 100

Applicable Treatments: `THINNED` treatments

Used in `tgdb1r00`**:** Yes

Description: The percentage of the basal area per hectare (acre) removed from a forest stand during a thinning operation. The interval width of 10% is used because the percentage of basal area removed during a particular thinning event is expected to be less than 50% of the standing basal area, and 10% interval widths should provide a reasonable resolution.

`YEARS_SINCE_TREATMENT`

Index parameter type: `INTERVAL`

Index parameter interval width: 4%

Valid index parameter values: Any numeric value ≥ 0

Applicable Treatments: All treatment types except the `UNTREATED` type

Used in `tgdb1r00`**:** Yes

Description: The number of years since a treatment was performed. The interval width of four (4) years is used to avoid possible aliasing problems caused by an interval width that is too small, given the remeasurement intervals which ranged from two (2) to eight (8) years. This interval width also takes into account the fact that there is some variation in the early effects of treatments.

Chapter 3

METHODS, DATA, TESTING AND VALIDATION

This chapter describes the theoretical and practical motivation for the tree list generation methodology proposed in Chapter 1 and implemented in the tree list generation database system described in Chapter 2. Following the description of the tree list generation database methodology is a brief description of a set of stand measurement data from Oregon, Washington, and southern British Columbia that were used to populate a tree list generation database, `tgdb1r00`, created using the methods described and the tree list generation database software. Next, the effectiveness of the tree list generation methodology is demonstrated by using the tree list generation database `tgdb1r00` and the tree list generation database software to generate simulated stand measurements or tree lists which mimic each actual stand measurement. The actual and simulated stand measurements are then compared using several criteria. The actual stand *vs.* simulated stand comparisons are the primary testing and validation procedures for the tree list generation database methodology, yielding empirical misclassification, or error, rates as an indication of overall performance. Finally, the chapter concludes with a summary of the tree list generation database validation results, some caveats for its use, and several other considerations.

3.1 Methods

In the abstract, there are five requirements for generating simulated stands or tree lists. First, is a stand classification procedure which partitions forest stands into clusters or groups based on their similarity, as determined by some set of stand at-

tributes. Second, is a stand selection procedure which identifies a group of stands, or several groups of stands, from a stand classification that have similar attributes to a specified stand for which a tree list is desired. Third is a method for representing a forest stand encompasses the wide variety of size, age, and species combinations that are found in actual stands. Fourth is a method for generating individual trees from the forest stand representation. Finally, there must be a wide variety of measurement data available to *calibrate* a tree list generation procedure. The fifth requirement is taken for granted, and the first four must be specified to produce a tree list generation methodology.

Given the broad scope of the abstract tree list generation problem, there are many possible alternatives for specifying the first four requirements in a solution. To narrow the scope of the simulated stand or tree list generation problem, several design criteria have been imposed on the stand classification and stand selection procedures and the stand representation and tree generation procedure. The design criteria are intended to address shortcomings of the classical approach to the problem, the Weibull parameter recovery approach, or to make the most effective use of the stand measurement data that are available. The following design criteria were used to help guide the development of the tree list generation methodology.

- The utility of the stand measurement data used to calibrate the tree list generation procedure should be maximized. Tree data are expensive to collect, and should therefore be put to the best possible use.

- Individual trees in a simulated stand or tree list should be generated as multidimensional objects directly. A tree is generally represented by a number of attributes, e.g., height, DBH, and species, and it should be possible to use more than just diameter, for example, when generating a tree list.

- The stand representation should be flexible enough to represent multimodal

and relatively flat size distributions, as well as unimodal size distributions, and arbitrary species compositions. Forest stands, including plantations, may obtain a wide variety of size distributions and species compositions.

- To the extent possible, simulated trees should be physically realizeable, that is, it should be possible to find an actual stand where a simulated tree could have occurred. Thus the tree list generation methodology should only generate *realistic* trees.

- Multiple treatments must be supported in a consistent manner. The self–consistency of a tree list generation database and a tree list generation procedure across treatments must be guaranteed.

- The addition of new stand measurement data should be easy to perform. Stand measurement data are collected on a regular basis, and it should be possible to easily update a tree list generation database as new data are collected.

- The addition of new stand measurement data should not modify the existing stand classification structure, except possibly by the creation of a new stand class. A particular tree list generation database and a tree list generation procedure must remain consistent, in terms of the simulated stands produced, as new data are added.

- The simulated stand or tree list generation framework should be as similar as possible to the standard framework of psuedorandom number generation. The simulated stand generation procedure should be placed firmly in a well understood computational and conceptual setting.

The stand classification and stand selection procedures, and the stand representation and individual tree generation procedure, selected to solve the tree list generation

problem follow a nonparametric statistical theme. The stand classification procedure is based on a multidimensional histogram, which is a consistent probability density function estimator. The stand selection procedure is based on a nearest neighbor approach, selecting the multidimensional histogram bins nearest to a desired stand description. The stand representation is an implicitly defined, i.e., data based, mixture distribution. The tree generation procedure is also based on a nearest neighbor approach using actual tree data. The nonparametric methods used allow the stand measurement data to *speak for itself* in determining a stand classification and in generating individual trees, as well as providing a great deal of flexibility to meet the design criteria.

There are three additional reasons for the nonparametric theme. First, traditional statistical approaches to these types of problems generally require a wide variety of assumptions involving the point estimates of parameters which characterize a generally one dimensional, unimodal distribution assumed to underlie a phenomenon or process, e.g., the Weibull or normal distribution. In addition, they require some sort of regression procedure and all of its requisite assumptions, e.g., the normality of the distribution of residuals. Second, nonparametric statistical techniques for solving these types of problems require far fewer assumptions than their classical counterparts. Third, the tremendous increases in computer processing power and storage capacities, coupled with significant reductions in cost and the ubiquitous presence of desktop computers, have made the storage and processing of large amounts of data possible in real time, or nearly real time, i.e., seconds or minutes rather than minutes to hours or hours to days.

The nonparametric approach to the tree list generation problem and the ready availability of high speed, large capacity computers provides the freedom to focus attention on the essential nature of the problem with a minimum number of necessary statistical or modeling assumptions. The nonparametric approach has the added benefit that it will work equally well for managed, planted forest stands and natural

forest stands. With this in mind, the remainder of this section provides details on the stand classification and stand selection procedures and the stand representation and tree generation procedure that have been used to develop the tree list generation database methodology and its supporting software, described in Chapter 2. When reading the tree list generation database component descriptions, keep in mind the two fundamental assumptions **FA1** and **FA2**: forest stands with similar attributes are similar, and sampled forest inventory data may be extended to a larger forested area as individual trees, respectively. The fundamental assumptions may be found at the end of Section 1.1.

3.1.1 Stand classification procedure

The tree list generation database stand classification procedure is based on a multidimensional histogram whose dimensions are defined by a particular set of stand attributes. The multidimensional histogram bin boundaries define the set of stand classification rules. Individual stands are classified as belonging to a particular multidimensional histogram bin, with other similar stands being assigned to the same bin. This binning of similar stands plays the same role as the individual parameter regression models, and their requisite assumptions, in the classical parameter recovery approach, without the problems associated with regression in many dimensions, e.g., the flatness of regression surfaces and the large data requirements, commonly referred to as the curse of dimensionality.

The multidimensional histogram classification or partitioning of the stand measurement data is performed using an *a priori* set of selected stand attributes, called stand index parameters, and their computed values for each set of stand measurement data. This type stand classification procedure provides a basic aggregation of the data, grouping stands which fall into the same multidimensional histogram bin together. Aggregating the data in this manner produces an *average* or *canonical* stand, evening out any eccentricities of the individual stands sampled for a particular stand

condition. The aggregation of the individual stand measurements in this way has the added benefit of reducing the size of the search index when performing a search to select stands similar to a specified stand description.

The stand index parameters may be continuous variables, e.g., QMD or mean tree height, or they may be categorical variables , e.g., stand type or stand origin. The histogram bins for the continuous stand index parameters are defined by specifying an appropriate bin width for that dimension of the multidimensional histogram, and they are called *interval* index parameters. The histogram bins for the categorical index parameters consist of a small set of distinct, fixed values, and they are called *fixed* index parameters. The fixed stand index parameters do not actually form bins, being single points, but they still partition the stand measurement data into separate classes, and they will be referred to as forming histogram bins.

The multidimensional histogram was chosen as the stand classification procedure for four reasons. First is its simplicity, both conceptually and computationally. Second, the statistical properties of the histogram as a probability density function estimator are well understood in a variety of different contexts [28, 32, 25, 10, 9, 24], including the consistency of the histogram and its asymptotic properties as the sample size approaches infinity. Third, the limited range of stand attribute values possible for forest stands, whether natural or managed, lends itself to a sequence of simple classification rules in each variable for descriminating between similarand dissimilar stand types. The histogram provides such a sequence of classification rules via its bin boundaries. Fourth, new stand measurement data may easily be added using this type of classification method, and only one histogram bin is affected by the addition of the new data: an existing bin, if the new stand attributes are similar to at least one stand that is already classified, or a new bin, if the new stand attributes are not similar to any previously classified stands.

The grouping of similar stands into the same multidimensional histogram bin is not necessary for the proposed tree list generation algorithm. An alternative would

be to simply store and index each stand individually by its specific attributes. The stand selection procedure defined in Section 3.1.2 is unchanged, except that stands are selected based on their individual attributes rather than selecting stands based on the centers of the multidimensional histogram bins which contain them. Additionally, the similarity function may be changed to a pure Euclidean distance function, rather than the weighted Euclidean distance function described in the next section. If there is a single stand in each histogram bin and the stand selection procedure below is used with an Euclidean distance function to choose the single closest stand for the desired stand attributes, the most similar neighbor method for selecting simulated stands results if the selected stand is simply copied [29].

3.1.2 Nearest neighbor stand selection procedure

The tree list generation database stand selection procedure is based on the idea of finding the multidimensional histogram bins that are similar to a requested stand description. The similarity is based on the set of stand index parameters in use and a similarity function. The similarity function is the bin width weighted Euclidean distance from the desired stand description, defined by the values of its stand index parameters, to the centers of the histogram bins in a tree list generation database stand classification. Let x be the vector of n stand index parameter values for the desired stand. The similarity score for determining the nearest neighbors is computed using the formula

$$S_j(x) = \sum_{i=1}^{n} w_i(x_i - c_{ij})^2, j = 1, 2, \ldots, N,$$

with

$$w_i = \begin{cases} \frac{1}{h_i} & \text{if } x_i \text{ is an interval valued index parameter} \\ 10000 & \text{if } x_i \text{ is a fixed valued index parameter, e.g., species} \end{cases}$$

where h_i is the multidimensional histogram bin width for an interval index parameter, the values for the c_{ij} are the histogram bin centers, and N is the number of multidi-

mensional histogram bins in a tree list generation database for a specified treatment. The scores $S_j(x)$ are ordered from smallest to largest. The five (5) or fewer multidimensional histogram bins having similarity scores less than five (5.0), $S_j(x) <= 5.0$, are selected to be used as the basis for generating a simulated stand or tree list using the procedure defined in Section 3.1.4. The multidimensional histogram bins used are those most similar to the desired stand description.

Once appropriate stands, multidimensional histogram bins, have been selected based upon their similarity scores, their nearness to the desired set of stand attributes, the individual tree data for each selected stand, histogram bin, are combined to form a lookup table of tree DBH and height measurements and tree species. This table is then treated as representing a *canonical stand* that is representative of the desired stand, and is used to randomly generate the individual trees for a simulated stand.

A similarity score in excess of 10000 indicates there were no matching stands in the tree list generation database, and no tree list can be generated. This can only occur if a value for a fixed index parameter is supplied and there are no corresponding values for that index parameter in a tree list generation database. This condition will only occurs if there are no data for the desired stand condition. The values of the maximum acceptable similarity score and the maximum number of qualifying stands are somewhat arbitrary, and any reasonable values may be used for the the stand selection criteria. The values of five (5.0) for the maximum similarity score and five (5) for the maximum number of similar stands were determined to be reasonable through trial and error with a visual assessment of the quality of the simulated stands. Values in the range of five(5) to ten (10) are generally considered to be reasonable for the number of nearest neighbors to include for smoothing [31]. These values may be modified when using the `TGRAND` command line options `-nn_max_score` and `-nn_max_stands`, respectively.

If the maximum number of qualifying stands is set to one, using the command line option `-nn_max_stands`, or there is only a single multidimensional histogram

bin with a similarity score less than the default value of five (5.0), or some other maximum similarity score defined by the command line option `-nn_max_score`, this stand selection procedure is like the most similar neighbor method [29]. The proposed stand selection procedure is identical to the most similar neighbor method if there is a single stand in each multidimensional histogram bin and an Euclidean distance similarity function is used.

3.1.3 Stand representation

A forest stand may be composed of many tree species, with one or more cohorts, or age classes, within each species, and a variety of tree sizes for each species or cohort grouping. The inherent variability in a forest stand is nicely captured by the notion of a *finite mixture density* or *mixture distribution* [33, 2, 26, 3], and this is the stand representation used for the tree list generation database. A finite mixture density is a probability density function of the form

$$p(x \mid (\alpha, \phi)) = \sum_{i=1}^{m} \alpha_i p_i(x, \phi_i))$$

where each $\alpha_i > 0, i = 1, 2, \ldots, m$, $\sum_{i=1}^{m} \alpha_i = 1$, and where each function p_i is itself a probability density function parameterized by a scalar or vector ϕ_i [33, 26, 2]. The function p would represent the overall structure of a forest stand, with each function p_i representing, possibly, a different species, cohort, or size class and the α_i representing the proportion of the stand associated with that particular species, cohort, or size class. The individual probability density functions p_i may be the same e.g., the Weibull density function [2, 3], or the normal density function, but this is not necessary. Each p_i could be specifically selected to suit a particular stand attribute. Additionally, the p_i need not be parametric density functions, like those mentioned, they may be nonparametric density function estimates [28, 9, 10, 26, 24, 25] created for a particular stand component. The mixture distribution is also well suited for representing forest stands since the general problem of estimating its parameters

is well understood [26], as is the problem of generating random numbers from a particular mixture distribution.

Given a mixture density as the representation of a forest stand, it may be specified in two ways: explicitly or implicitly. If the mixture distribution for the stand p is specified explicitly, there are again two approaches: a parametric approach and a nonparametric approach. For the parametric approach individual probability density functions p_i must be chosen, and their individual parameters ϕ_i and the mixing coefficients α_i estimated [26]. Once all of the parameters for the mixture distribution have been estimated, it may be used to generate a tree list. The traditional Weibull parameter recovery method used for diameter distribution modeling is a form of the parametric approach, though with additional assumptions and data processing steps [2, 3]. For the explicit nonparametric approach, the data for a stand are used to estimate the parameters for a particular type of nonparametric probability density function estimator [28, 32, 25, 10, 9, 24]. The nonparametric density function is then used to generate a tree list for a particular stand description. A combined approach using both parametric and nonparametric probability density functions is also possible.

The implicit approach for specifying a mixture density to represent a forest stand is also a nonparametric approach. The individual tree data, DBH, height, and species for a stand are stored and then used with an appropriate procedure to generate random trees having the proper statistical characteristics for the stand. This is done either by forming an explicit, local representation for the mixture density p, using local nonparametric density estimation methods, or by using the data directly to generate simulated trees. The latter approach is the preferred method, making the most effective use of the tree measurement data, provided a procedure for generating the individual trees is available. Section 3.1.4 describes one such method.

The implicit mixture density representation of a stand was selected for use in the tree list generation database. The implicit mixture density approach has the

advantage that no parameter estimation must be performed to determine a stand description; the data implicitly define a stand. Given a data based tree generation procedure there is no actual effort that must be expended to use the implicit stand representation, it simply provides a convenient and powerful conceptual model for describing a forest stand. The drawback to the approach is that all of the individual tree data must be available in order to generate simulated trees. This is not, however, a major constraint given the inexpensive, large capacity storage available for desktop computers.

So, given the implicit mixture density based stand representation, simulated trees may be generated by first selecting an appropriate set of individual trees, and then using a data based tree generation procedure. An appropriate set of individual trees, defined as compatible DBH and height measurements with tree species, are selected, e.g., using the procedures from Section 3.1.2. The tree generation procedure described in Section 3.1.4 may then be used with the selected trees to directly generate simulated trees.

3.1.4 Tree list generation procedure

The tree list generation database tree generation procedure is based on a method of generating simulated data vectors having the characteristics of a particular data set [31]. The tree generation procedure is first presented in its generality, and then the modifications necessary to specialize the procedure for the tree list generation problem are presented. The description of the algorithm is summarized here, with an emphasis on its practical and computational aspects. For a complete discussion of the procedure, the reader is referred to the original paper [31].

Let $X_j \in \Re^k$, $j = 1, 2, \ldots, n$ be a random sample of size n from an unknown multidimensional probability density function. Perform a variance normalizing transformation on the data vectors, $X^t\Sigma^{-1}$, where $X^t = \{X_j^t\}$ is the $n \times k$ data matrix,

and

$$\Sigma_{ii}^{-1} = \frac{1}{s_i},\ i = 1, 2, \ldots, k$$
$$\Sigma_{ij}^{-1} = 0,\ i, j = 1, 2, \ldots, k, i \neq j,$$

where s_i is the standard deviation of data dimension i, to account for any differences in variance among the k coordinate dimensions. Next select at random one of the n data points, say X_r, and determine its m nearest neighbors. The nearest neighbors are determined by computing the Euclidean distance from X_r to X_j, $D_{rj} = \sqrt{(X_r - X_j)^t (X_r - X_j)}$, for $j = 1, 2, \ldots, n$ and then selecting the m smallest distances. The initially selected data vector X_r is always selected as one of the nearest neighbors; it is the closest vector ot itself with an Euclidean distance of zero. For convenience, relabel the m nearest neighbors $Y_j, j = 1, 2, \ldots, m$.

The m nearest neighbor data vectors, Y_j, are transformed by subtracting their sample mean, $\bar{Y} = \frac{1}{m} \sum_{j=1}^{m} Y_j$, yielding the data vectors $Y_j' = Y_j - \bar{Y}$, $j = 1, 2, \ldots, m$. Next a random sample $u_1, u_2, \ldots, u_m$ is generated from the uniform distribution

$$U\left(\frac{1}{m} - \left[\frac{3(m-1)}{m^2}\right]^{\frac{1}{2}}, \frac{1}{m} + \left[\frac{3(m-1)}{m^2}\right]^{\frac{1}{2}}\right),$$

selected to ensure appropriate properties for the first and second moments, the mean and covariance, of the simulated data vectors. The uniform random numbers u_j are combined with the transformed vectors Y_j' to get a simulated data vector

$$Y' = \sum_{j=1}^{m} u_j Y_j'$$

which is transformed into its original coordinates

$$Y = \Sigma \left(Y' + \bar{Y}\right)$$

to obtain a simulated data vector that is representative of the original multivariate distribution of the data X_j, $j = 1, 2, \ldots, n$. This procedure is repeated for each desired simulated data vector, selecting a new initial data vector X_r with replacement.

The random vectors Y generated by this procedure are uncorrelated and they approximately reproduce the covariance structure of the original data $X_j \in \Re^k$, $j = 1, 2, \ldots, n$ [31].

A theoretical, asymptotic dependence between m, the smoothing or averaging parameter, and the sample size n exists, as for the sample size and the bin width or number of bins for histograms and a variety of other nonparametric probability density estimators [32, 28, 25, 10, 9, 24]. For practical purposes a fixed value of m may usually be chosen without incurring a significant penalty. Empirical results support this conclusion, and a value of $m = 10$ has been shown to give good results [31]. The value for $m = 10$ has therefore been chosen for use in the tree list generation algorithm of the tree list generation database.

A tree list is a three dimensional vector, with two continuous dimensions, tree DBH and height, and one discrete or categorical dimension, tree species. To generate a simulated tree list, an actual tree list, i.e., a list of actual tree DBH and height measurements with species, that is presumed to be representative of the desired simulated tree list is necessary. The DBH and height components of a tree list are considered as a two (2) dimensional vector and correspond to the X_j in the random vector generating algorithm. The species of a simulated tree is determined by the species of the first randomly selected tree, corresponding to the X_r in the random vector generating algorithm, for each simulated tree.

The actual DBH height, and species values may be obtained by the procedure described in Section 3.1.2 or some other procedure. Let this actual tree list contain n trees. The tree list generation procedure is then given by the following algorithm.

Step 1 Generate a random integer r between 1 and n. This selects one of the n trees from the canonical stand or actual tree list.

Step 2 Assign the species of the tree selected in Step 1 to the tree being simulated.

Step 3 Use the random vector generation procedure with the actual DBH and height measurement data to generate a simulated pair of DBH and height measurements.

Step 4 If either of the simulated DBH or height measurements is not positive, repeat Step 3 until both of the simulated values are positive.

Step 5 Repeat Step 1 through Step 4 until all desired trees have been generated.

The species of a simulated tree cannot be generated using the random vector generation procedure since the procedure may only be applied to continuous variables. This necessitates the two part procedure outlined by Step 1 and Step 2. Step 4 simply guarantees that simulated DBH and height measurements will always be positive.

Two modifications of the tree list generation algorithm defined by Step 1-Step 5 will prove useful. These modifications address the issue of generating a tree list that contains only one species, that is, a tree list that is 100% pure, and the issue of using estimated tree heights to generate simulated trees. The need for the first modification should be readily apparent. The need for the second modification may need a bit of motivation.

The majority of individual tree heights in a sample are not measured, but are estimated from height-diameter relationships determined by a small set of trees with measured heights. These estimated tree heights typically follow a smooth curve, giving the estimated tree heights less variability relative to the actual variability of tree heights observable for a particular DBH. Given this reduction in variability, a method for adding variability or jittering the simulated tree heights is desirable to more accurately represent the possible natural variability of the tree heights for a given DBH.

The first modification to the tree list generation algorithm allows the generation of a 100% pure stand, and is controlled by the value of the `RANDOM_MF_PURE` keyword

in a stand description file supplied to TGRAND. See Section 2.11.2 for details. For a stand with a type of PURE_DF or PURE_WH, all tree records for species other than the dominant tree species are removed from the actual list of trees before continuing with the algorithm. Stand types other than PURE_DF and PURE_WH are unaffected by this procedure. This introduces a Step 0 to the algorithm.

Step 0 If a 100% pure stand of trees is to be generated, all of the tree records for tree species other than the dominant, desired tree species are removed from the actual tree list. Proceed to Step 1.

The second modification to the tree list generation algorithm adds a small amount of random jitter to each simulated tree height in an attempt to more adequately represent the natural variation in tree heights in a simulated stand. The addition of jitter to the tree heights is controlled by the value of the RANDOM_MF_JITTER keyword in a stand description file supplied to TGRAND. See Section 2.11.2 for details. The ability to jitter the simulated tree heights is important because most tree heights are estimated from height-diameter models, and hence follow a smooth curve, giving tree heights very little variability relative to the actual variability in tree heights for a particular diameter. The jitter is computed using the specified stand age, from a stand description file, and a simulated tree height to compute a mean annual height increment (MAI) for each tree. A uniform random number u is then generated from the interval $(-2\ h_{\mathrm{MAI}}, 2\ h_{\mathrm{MAI}})$, and added to the simulated tree height. This introduces a Step 3a to the algorithm.

Step 3a If a simulated tree height h is to be jittered, then

3a-1 Compute the mean annual height increment using the simulated tree height h from Step 3 and the stand total age a, $h_{\mathrm{MAI}} = \frac{h}{a}$.

3a-2 Generate a uniform random number u from the interval $(-2\ h_{\mathrm{MAI}}, 2\ h_{\mathrm{MAI}})$.

3a-3 Let $h = h + u$.

3a-4 Proceed to Step 4.

A nonparametric procedure for generating simulated tree lists has been proposed. The underlying representation of a forest stand is that of a mixture distribution. The mixture distribution is represented as a multidimensional histogram and used as a stand classifier. A nearest neighbor stand selection procedure selects stands that are most similar to a specified set of stand attributes. These stands are then used to generate a simulated stand having attributes similar to those desired. Two modifications to the basic procedure are also provided. These permit the generation pure or single species simulated stands, and the generation of simulated stands with jittered tree heights. The tree list generation procedures outlined have been implemented in Fortran 90/95 and will be applied to a large data set that is representative of Douglas-fir and western hemlock plantations in the Pacific Northwest.

3.2 Data

The tree list generation database methodology is intended to provide a means for generating simulated stands of trees, or tree lists, that are representative of the Pacific Northwest region west of the Cascade Mountains, and extending from southern Oregon to southern British Columbia. Figure 3.1 presents a map of the appropriate region, showing the locations from which data were obtained. To achieve the goal of regional tree list generation, stand measurement data were provided by the organizations listed in Table 3.1.The data consisted of stand measurements from pure Douglas-fir stands, pure western hemlock stands, and stands with mixtures of Douglas-fir and western hemlock.

The ability to generate simulated stands or tree lists for both untreated stands and thinned stands was of primary interest for the tree list generation database. Individual tree data, consisting of compatible DBH and height measurements with an

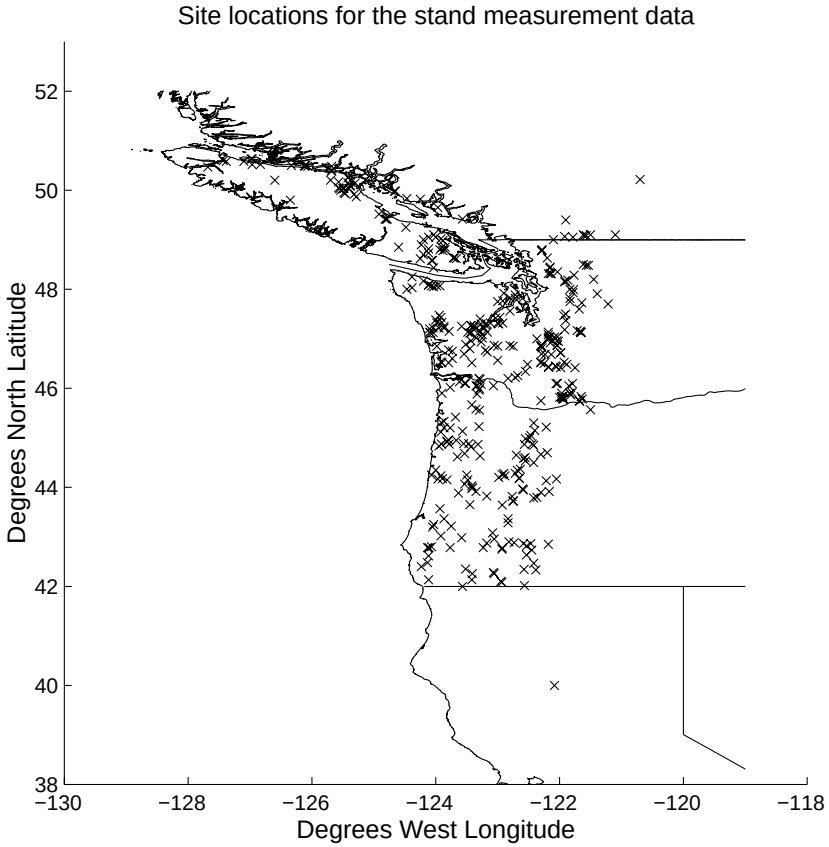

Figure 3.1: Tree measurement data locations for the untreated and thinned stand data used to create the tree list generation database `tgdb1r00`.

Table 3.1: List of data sources for the tree list generation database.

Data Sources
British Columbia Ministry of Forests
Washington State Department of Natural Resources
Canadian Forest Service
Oregon State University, Corvalis, Oregon
USFS Pacific Northwest Research Station
Port Blakely Tree Farms
Regional Forest Nutrition Research Project (RFNRP), SMC
Stand Management Cooperative (SMC), University of Washington
Weyerhaeuser Company

indication of tree species, for these two treatments were extracted from the data sets provided and were used to create the tree list generation database `tgdb1r00`. Only live trees were used to create the tree list generation database. Table 3.2 contains the breakdown of stands by stand type and treatment. For a more detailed summary of the data used to create the tree list generation database `tgdb1r00` see Section 2.10.

Table 3.2: Stand type summary for all stands used to create the basic tree list generation database. Percentages are rounded to the nearest 0.1%, so the row and column percent totals may not agree exactly.

	Untreated stands		Thinned stands		Total	
Stand type	Count	Percent	Count	Percent	Count	Percent
PURE_DF	3404	35.3	3431	35.6	6835	70.8
PURE_WH	800	8.3	480	5.0	1280	13.3
DF_DOMINANT	691	7.2	417	4.3	1108	11.5
WH_DOMINANT	178	1.8	86	0.9	264	2.7
MIXTURE	136	1.4	26	0.3	162	1.7
Total	5209	54.0	4440	46.0	9649	100.0

Given the wide variety of data sources, a myriad of data collection and sampling strategies were likely employed, with differing assumptions, and a variety of unknown height-diameter relationships would have been used to estimate most of the individual tree heights. If the data collection histories, data sampling schemes, and other statistical procedures were known, it would be nearly impossible to reconcile all of the discrepancies that are undoubtedly present, and it would be impossible without that information, which is the case with the majority of this data. Therefore, no attempt was made to address the statistical compatibility of the sampling strategies or other statistical procedures that were employed historically for these data sets, and the data were assumed to be compatible and used *as is*. To help ensure the reliability of the tree list generation database, only trees from sample plots that were at least 0.0405 ha (0.10 ac) were used.

The stand measurement data supplied contained breast height ages, but stand

total ages were desired for the tree list generation database. Stand total ages for Douglas-fir were computed from stand breast height ages using Bruce's equations [4]. Stand total ages for western hemlock stands were computed in the same manner, for lack of a breast height age to total age conversion formula for western hemlock. The breast height age to stand total age conversion equation used was:

$$A_T = A_{BH} + 13.25 - \frac{SI}{20},$$

where A_T is stand total age, A_{BH} is breast height age, and SI is site index at age 50.

For pure Douglas-fir or pure western hemlock stands, a species specific 50 year site index value was usually provided. For mixed Douglas-fir and western hemlock stands, if two site index values were provided, the site index value used was the value for the species with the greater percentage of stand basal area. If a single species specific site index value was provided for a stand, and that species had the lower percentage of stand basal area a site index conversion equation was used to convert the given site index value to a site index value for the more dominant species [21]. The equation

$$SI_{WH} = 3.2808 \left(0.432 + 0.899 \frac{SI_{DF}}{3.2808}\right)$$

was used to convert a Douglas-fir site index value to a western hemlock site index value, and the equation

$$SI_{DF} = 3.2808 \left(0.480 + 1.11 \frac{SI_{WH}}{3.2808}\right)$$

was used to convert a western hemlock site index value to a Douglas-fir site index value. In both equations the site index values SI_{DF} and SI_{WH} are assumed to be in feet and for a reference age of 50 years. If the site index values are in meters, replace the value 3.2808 in the equations by the value 1. If no site index value was provided for a stand, that stand was removed from the data set and not considered further.

An indication of stand origin was provided for each stand. Stand origins were one of natural regeneration, seeded, planted, or unknown. The tree list generation

database provides two stand origin types: natural and planted. Stand origins other than these two values must, therefore, be appropriately converted before the data for a stand may be used in a tree list generation database. Stand origins from the data that were either unknown or seeded were assumed to be naturally regenerated. Any stand having an unrecognized stand origin was removed from the data set and not considered further.

For each thinned stand, pre-thinning density, pre-thinning basal area, and the years since thinning were provided in addition to the post-thinning list of tree measurements and species. No pre-thinning list of tree measurements and species was provided. No attempt was made to simulate a pre-thinning stand from the information provided; only the post-thinning tree information was used for the tree list generation database. The lack of a complete pre-thinning list of tree measurements and species had the effect of limiting the years since treatment for the tree list generation database to a value of zero for many of the thinned stands, regardless of the number of prior thinnings. Both pre-thinning and post-thinning lists of tree measurements and species should be kept in the future to alleviate this type difficulty.

3.3 Testing and validation

The testing and validation of the tree list generation database has two components. First is verifying that the tree list generation database software, the executable programs and through them the subroutine library, work as expected. Second is verifying that a tree list generated from a tree list generation database is a reasonable approximation of the stand that it was generated to mimic. The primary emphasis for the testing and validation of the tree list generation database is the second component, since this is of greatest importance for determining whether the overall tree list generation methodology is valid. Testing of the executable program interfaces and the subroutine library was accomplished by using the tree list generation database

software to generate simulated tree lists for the validation of the tree list generation methodology. For more information on the software testing procedures and philosophy see Section 3.3.5.

To determine whether a tree list generated using a tree list generation database is a reasonable approximation to an actual stand, as described by the stand characteristics specified in a stand description file, the notion of *reasonable* must be defined quantitatively. A standard approach for this type of problem is to place it in the context of determining the goodness of fit between an actual data set, or collection of data sets, and simulated data generated to mimic the actual data set, or data sets. This is the general approach taken here [3, 27, 1].

Specifically, for each actual stand measurement, a simulated stand measurement was generated to mimic its attributes using a tree list generation database. The actual and simulated stand measurements were then compared using one or more goodness of fit criteria. An error rate, or misclassification rate, for the comparison of actual and simulated stands is determined from the goodness of fit testing methods and criteria applied. This provided an indication of the robustness of the tree list generation database methodology at the stand level. The overall robustness of the tree list generation methodology, or the regional accuracy, was assessed by performing a simple linear regression analysis comparing the actual and simulated QMD and mean height values for each stand, and by comparing the overall distributions of these two stand attributes for the actual and simulated stands.

The stand attributes or characteristics used to generate the simulated tree lists are the stand index parameters defined for a particular tree list generation database. The goodness of fit testing methods and criteria used are described in Section 3.3.1. The results of the goodness of fit testing analyses for untreated stands and thinned stands follow in Section 3.3.3 and Section 3.3.4, respectively. The goodness of fit testing results presented are for the tree list generation database `tgdb1r00`. See the specific results sections for the list of stand attributes used as the tree list generation

database stand index parameters.

3.3.1 Goodness of fit testing methods and criteria

The classical goodness of fit testing scenario is based on determining whether a set of observed data values are in agreement with a set of expected data values. The degree of agreement may be defined in terms of point estimates of the parameters of a distribution or density function derived from the data values in each data set, such as a mean and variance, or in terms of the actual distributions of the data values in each data set. In the first case, commonly called parametric testing, knowledge of an underlying probability density function is assumed, or known, *a priori*, and the goodness of fit testing is reduced to a comparison of estimated parameters for the underlying probability distribution that are derived from the observed data values and the expected data values. An example of the parametric approach is the t-test [1]. In the second case, knowledge of an underlying probability density function is not necessarily assumed, or known, and the goodness of fit testing is based on a comparison of the distributions of the data values. Examples of this approach are Pearson's chi-square test for categorical data and the nonparametric Kolmogorov-Smirnov test for continuous or discrete ordinal data [1].

The goodness of fit testing procedures used to compare the actual stands and the simulated stands generated using the tree list generation database `tgdb1r00` must emphasize the distributions of the tree DBH and height measurements, both continuous variables, and the species composition or distribution, a categorical variable. The comparison of the distributions, rather than point estimates of parameters, is essential for determining whether the tree list generation methodology of the tree list generation database methodology is appropriate. The tree size distributions and the species composition within a forest stand are its fundamental measured components, and are used for the goodness of fit comparisons. Any other type of comparison must make less effective use of the data due to information lost by data aggregation, e.g.,

binning continuous data or reducing a data set to a few parameters such as a mean and variance.

The classical methods for performing goodness of fit tests, whether producing point estimates or comparing distributions are, unfortunately, not applicable to, or are of rather limited value for, the tree list generation database goodness of fit testing. Point estimates such as the mean and variance, as used in a t-test, for DBH or height distributions generally prove to be inadequate when describing forest stands. The distributions of both DBH and height are frequently not unimodal and may be multimodal or highly skewed. In addition, the classical goodness of fit testing procedures may be inappropriate in situations where sample sizes are large, large sometimes being as small as 50 or 100 sample points, spuriously indicating that two data sets are different when they are in fact obtained from the indistinguishable underlying distribution [1, 6]. The nonparametric Kolmogorov-Smirnov goodness of fit test statistic comes closest to meeting the requirements for comparing the continuous DBH and height distributions, but it only takes into account the maximum difference between the two cumulative distribution functions [1], and not the differences across the entire range of the data sets being compared.

The Kolmogorov-Smirnov test statistic does, however, provide the motivation for another nonparametric test statistic that will overcome the spurious rejection problems and sample size limitations of the classical goodness of fit tests, such as the t-test and the chi-square test [1, 6], as well as emphasizing the complete distribution of the data in a particular data set. The test statistic is the integrated absolute error, or difference, between two functions f and g, which is defined by the integral

$$\mathrm{iae}(f, g) = \int_{-\infty}^{\infty} |f(x) - g(x)|\, dx.$$

The integrated absolute error is obviously related to the Kolmogorov-Smirnov test statistic, and has been used as an index for comparing diameter distributions [3, 27].

If the functions f and g are probability density functions, the integrated absolute

error computes the proportion of probability mass that the two functions do not have in common, or the percent difference in the area under the curves of the two probability density functions. Values of the integrated absolute error range from a minimum value of zero, indicating that the functions f and g are identical, to a maximum value of two (2), indicating that the functions f and g are completely different, i.e., have no overlap. To avoid any potential confusion in the interpretation of this test statistic, a transformation of the integrated absolute error is performed to obtain a statistic the same interpretation as the classical p-value, and the formula for the p_{iae}-value

$$p_{iae} = 1 - \frac{1}{2}\int_{-\infty}^{\infty} |f(x) - g(x)|\, dx,$$

is obtained.

The p_{iae}-value has a range from zero, indicating that two distributions are completely different, to one, indicating that the two distributions are identical. The p_{iae}-value may be interpreted as the proportion of probability mass that two probability density functions f and g have in common, or the percent similarity in the area under the curves of the two probability density functions. This interpretation of the p_{iae}-value is consistent with that of the classical p-value. Unlike most classical test statistics, the p_{iae}-value *automatically* works for multimodal distributions as well as unimodal distributions, both of which are common in forestry data. In addition, this type of test statistic is quite flexible, and may be used in a mixed form containing both continuous *and* discrete probability density functions without introducing any mathematical, statistical, or interpretive difficulties.

The p_{iae}-value formula has three primary variants which, span the range of data types from continuous data to categorical data, or continuous probability density functions to discrete probability density functions. The first p_{iae}-value variant is for continuous probability density functions f and g defined over all, or some subset, of

the real line, and is given by

$$p_{iae} = 1 - \frac{1}{2}\int_{-\infty}^{\infty} |f(x) - g(x)|\, dx. \tag{3.1}$$

This form of the p_{iae}-value statistic is appropriate for continuous data, such as tree DBH and height measurements.

The second p_{iae}-value variant is for discrete probability density functions f and g defined over a set of distinct values or sampling universe, $X = \{x_1, x_2, x_3, \ldots, x_n\}$ for a finite set of values or $X = \{x_1, x_2, x_3, \ldots\}$ for an infinite set of values, and is given by

$$p_{iae} = 1 - \frac{1}{2}\sum_{\text{all } x_i} |f(x_i) - g(x_i)|\,. \tag{3.2}$$

This form of the p_{iae}-value statistic is appropriate for categorical data, such as tree species.

The third p_{iae}-value variant is based on the histogram approximation to probability density functions, and is generally applicable to data from continuous distributions that have been binned to obtain bin frequencies. This variant of the p_{iae}-value formula has a strong resemblance to the formula for computing the chi-square statistic. Let a and b be the lower and upper limits for the range of the histograms for two probability density functions f and g, each having m bins, with bin width $h = \frac{b-a}{m}$, and bin boundaries defined by $x_i = a + ih, i = 0, 1, 2, \ldots, m$. Let f_i and g_i be the bin frequencies for the ith bin of the histogram approximations to f and g, respectively. The frequency values may be obtained as approximate areas under the probability density curves, or as the bin frequencies from two data sets with n_f and n_g data points, respectively, $f_i = \frac{n_{fi}}{n}$ and $g_i = \frac{n_{gi}}{n}$, where and n_{fi} and n_{gi} are the bin counts for the ith histogram bin for each function. The p_{iae}-value is then given by

$$p_{iae} = 1 - \frac{1}{2}\sum_{i=1}^{m} |f_i - g_i|\,. \tag{3.3}$$

The determination of a p_{iae}-value from two sets of continuous data or discrete data, one data set from f and one from g, using the p_{iae} formula for discrete or

categorical data, Equation 3.2, or the histogram based formula, Equation 3.3, is a straightforward application of the appropriate formula. Obtaining a p_{iae}-value from two sets of continuous data using the integral based p_{iae} formula, Equation 3.1, is only slightly more involved, consisting of two steps. The first step involves estimating a continuous probability density function for each of the two data set [28, 32, 24, 25, 10, 9]. The second step performs a numerical integration to obtain the p_{iae}-value. The probability density function estimator used in the first step is a nonparametric probability density estimation technique based on cubic B-splines [24, 25, 10, 9], though any appropriate nonparametric probability density estimation technique could be used in its place. The numerical integration performed in the second step is a straightforward midpoint based method, which may be obtained from any basic calculus text, for determining the area under a curve. Note that the p_{iae}-value formula given by Equation 3.3 is a special case of the p_{iae}-value formula in Equation 3.1, since the histogram is a nonparametric probability density function estimator, with the summation taking the place of the integration.

3.3.2 The p_{iae}-value as a test statistic

The p_{iae}-value is the test statistic that will be used to determine the empirical misclassification rates from the testing and validation of the tree list generation database methodology. The misclassification rates are derived from the individual tree DBH, height, and species goodness of fit comparisons between actual stands and simulated stands. The p_{iae}-value will be used most effectively as a test statistic if an appropriate p_{iae} cutoff value, or critical p_{iae}-value, can be determined. The critical p_{iae}-value will perform the same role as the critical value, or p-value, in a classical goodness of fit or hypothesis test.

The selection of a critical p_{iae}-value is performed in two stages. First, a comparison of the p_{iae}-value with the classical t-test is performed to provide insight into what the p_{iae}-value actually measures in a well understood statistical setting, and to show

that it has the correct properties to be used as a test statistic. This demonstrates the applicability of the p_{iae}-value as a test statistic. Second, a simulation of an actual sampling scenario is performed using the standard normal distribution. The simulation is used to characterize the distribution of the p_{iae}-value under a null hypothesis of identical distributions, and to guide the selection of an appropriate critical p_{iae}-value. A two sided t-test is also applied to the simulated data generating p-values to provide a comparison of the two statistics. This demonstrates the feasibility of using the p_{iae}-value as a test statistic. The two stages use the p_{iae}-value formula in Equation 3.1 and the nonparametric probability density function estimation and numerical integration procedures described. The behavior and properties of the other two p_{iae}-value formulae are similar to the one being tested.

To set the stage for the comparison of the p_{iae}-value and the t-test, a brief review of the classical hypothesis testing framework will prove useful, defining the terminology used in the comparison. In the classical hypothesis testing framework, an observation X is made, where X is assumed to be distributed according to P_θ, a distribution parameterized by some $\theta \in \Theta$, where Θ is the parameter space. Further, there are two possible states of nature described by the relations $\Theta_0 \subset \Theta$ and $\Theta_a \subset \Theta$, where Θ_0 and Θ_a are disjoint subsets of Θ, i.e., $\Theta_0 \cap \Theta_a$ is empty. The sets Θ_0 and Θ_a may be single points, intervals, or arbitrary disjoint subsets of Θ. The objective of testing an hypothesis is to determine whether the distribution of the sample X, P_θ, is described by $\theta \in \Theta_0$, called the null hypothesis H_0, or by $\theta \in \Theta_a$, called the alternative hypothesis H_a. Only one of the two states of nature $\theta \in \Theta_0$ or $\theta \in \Theta_a$ may be the true state of nature from which X was observed. The classical hypothesis testing framework is concerned with determining decision rules and test statistics which can distinguish between the two states of nature, selecting the most likely state of nature for the observation X.

Given a decision rule, a test statistic, and an observation X, a decision about which of $\theta \in \Theta_0$ or by $\theta \in \Theta_a$ is the true state of nature has four possible outcomes,

two correct outcomes and two incorrect outcomes, only one of which may occur at a time. The two correct outcomes are: $\theta \in \Theta_0$ and the true state of nature is described by Θ_0 and $\theta \in \Theta_a$ and the true state of nature is described by Θ_a. The incorrect outcomes indicate an error and are: $\theta \in \Theta_a$ and the true state of nature is described by Θ_0, a false rejection of H_0, and $\theta \in \Theta_0$ and the true state of nature is described by Θ_a, a false acceptance of H_0. The first incorrect outcome is called a Type I error and the second incorrect outcome is called a Type II error. The probability of a Type I error, $P(\text{Type I error})$, is commonly denoted by α, and is called the Type I error rate. The probability of a Type II error, $P(\text{Type II error})$, is commonly denoted by β, and is called the Type II error rate. The total error rate for a particular decision rule and test statistic, or hypothesis test, is given by $P(\text{Type I error}) + P(\text{Type II error}) = \alpha + \beta$. The Type I and Type II error rates, and hence the total error rate, are usually tightly linked to each other, to the underlying distribution P_θ, and to the distribution of the test statistic under the null hypothesis H_0.

When selecting a test, a test statistic and decision rule, to discriminate between the two states of nature, $\theta \in \Theta_0$ or $\theta \in \Theta_a$, it seems natural to choose a test which would minimize the total probability of error. Unfortunately, it is impossible, in general, to simultaneously have both $P(\text{Type I error}) = 0$ and $P(\text{Type II error}) = 0$; if $P(\text{Type I error}) = 0$ then $P(\text{Type II error}) = 1$ and if $P(\text{Type II error}) = 0$ then $P(\text{Type I error}) = 1$ [1]. So, for all practical purposes a statistical test must have some nonzero total error rate.

The composition of the total error rate for a test, in terms of the Type I and Type II error rates, may be controlled, to some degree, by placing the emphasis on one or the other type of error when using a particular test statistic. The standard approach in scientific studies is to emphasize the Type I error rate, selecting an *a priori* tolerance for false rejection of the null hypothesis H_0, commonly called the α-level of the test. The null hypothesis H_0 is rejected in favor of the alternative hypothesis $H_a$if the p-value of the test is less than the chosen α-level. For the classical statistical

tests, this approach allows some control over the Type II error rate as well: choosing a larger value for α reduces the value of β, and choosing a smaller value of α increases the value of β, demonstrating the coupling of the Type I and Type II errors.

Having completed the brief review of the classical hypothesis testing framework, the applicability of the p_{iae}-value as a test statistic will be demonstrated through a comparison with the classical t-test applied to two normal distributions with equal variances. In this setting, the t-test is the optimal statistical test for random samples drawn from the two normal distributions [1], and the exact probability of committing a Type I or Type II error may be computed. The p_{iae}-value is clearly related to the t-test error rates in this setting: it has the same theoretical value as the minimum total probability of error, or the total t-test error rate, $P(\text{Type I error}) + P(\text{Type II error})$, of the t-test.

The comparison the p_{iae}-value with the total error rate of the classical t-test was performed using eight tests. Each test compared two normal distributions with unit variance and different separations between their means. The first normal distribution in each comparison was a standard normal distribution, $N(\mu_1, 1) = N(0, 1)$, and the second was the normal distribution $N(\mu_2, 1)$, where μ_2 takes on the values 0, 1, 2, ..., 7. The first normal distribution is assumed to be the true, or reference, distribution for a t-test null hypothesis H_0 and the second distribution is the alternative distribution for a t-test alternative hypothesis H_a. Each test was performed by drawing a pair of random samples, one from each of the two normal distributions $N(\mu_1, 1)$ and $N(\mu_2, 1)$ for each value of μ_2, and computing the p_{iae}-value for that pair of samples. In each case, the random draw used a sample size of 1000 points, to ensure good nonparametric probability density estimates for each normal distribution, and was repeated three times to obtain a mean and standard deviation for each p_{iae}-value.

Figure 3.2 through Figure 3.4 present graphically the nonparametric probability density estimates for the testing procedure and $\mu_2 = 0$, 3, and 7, along with the true t-test total error rate and the mean and standard deviation for the each p_{iae}-value.

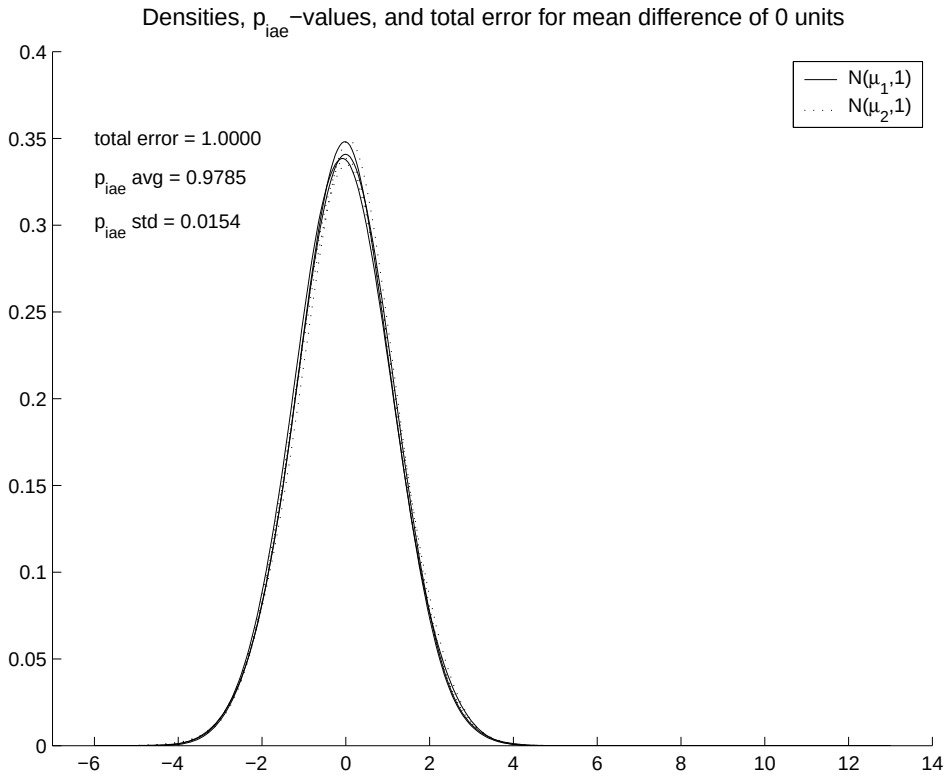

Figure 3.2: Nonparametric probability density estimates with the true t-test total error rate and the nonparametric p_{iae}-value (mean of three trials) from two normal distributions $N(\mu_1, 1)$ and $N(\mu_2, 1)$ where $\mu_1 = \mu_2 = 0$. The nonparametric probability density function estimates were generated from samples of 1000 points from each normal distribution.

Figure 3.5 and Table 3.3 present the true t-test total error rates and the p_{iae}-values for each of the eight tests. These figures and the table clearly indicate that the p_{iae}-value has the appropriate properties to be used as a test statistic, decreasing rapidly from a value near one for no difference in means to a value near zero for a difference in means of seven (7) standard deviations, and closely following the t-test total error rate. The small standard deviations for the computed mean p_{iae}-values in each comparison indicate that this type of test statistic is also quite robust, at least for the large sample sizes used.

Having shown that the large sample, or asymptotic properties, of the p_{iae}-value

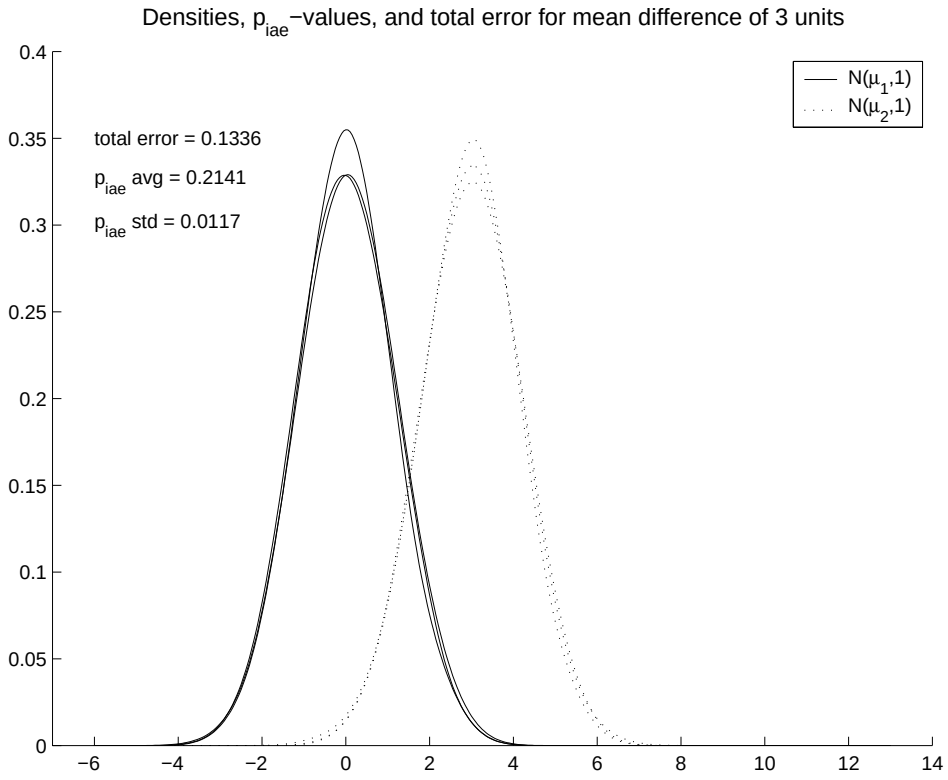

Figure 3.3: Nonparametric probability density estimates with the true t-test total error rate and the nonparametric p_{iae}-value (mean of three trials) from two normal distributions $N(\mu_1, 1)$ and $N(\mu_2, 1)$ where $\mu_1 = 0$ and $\mu_2 = 3$. The nonparametric probability density function estimates were generated from samples of 1000 points from each normal distribution.

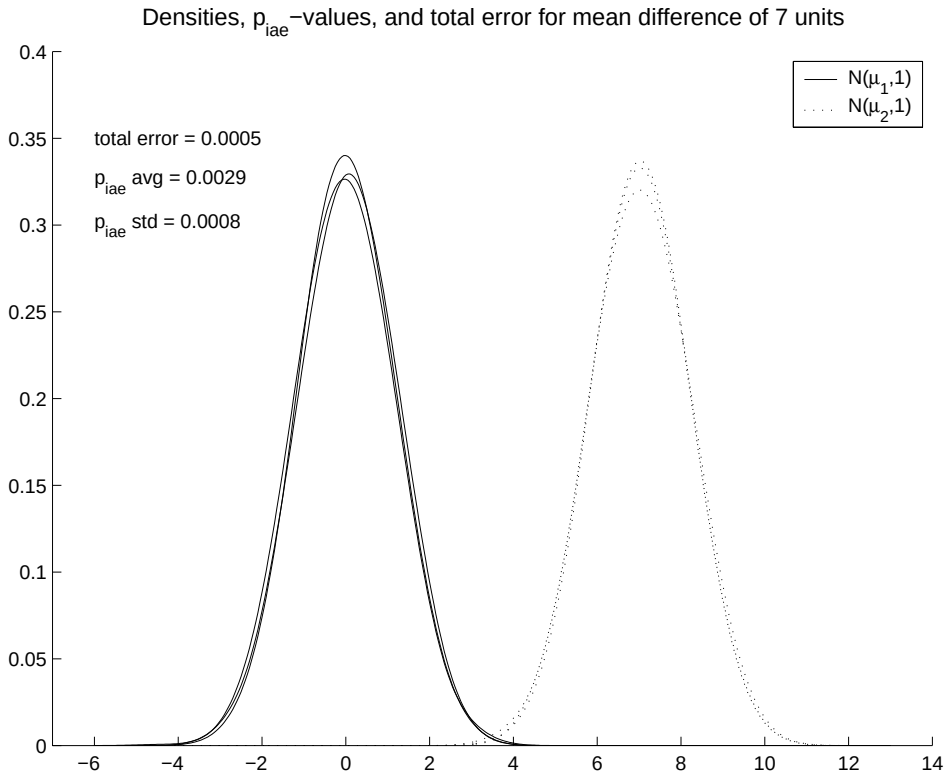

Figure 3.4: Nonparametric probability density estimates with the true t-test total error rate and the nonparametric p_{iae}-value (mean of three trials) from two normal distributions $N(\mu_1, 1)$ and $N(\mu_2, 1)$ where $\mu_1 = 0$ and $\mu_2 = 7$. The nonparametric probability density function estimates were generated from samples of 1000 points from each normal distribution.

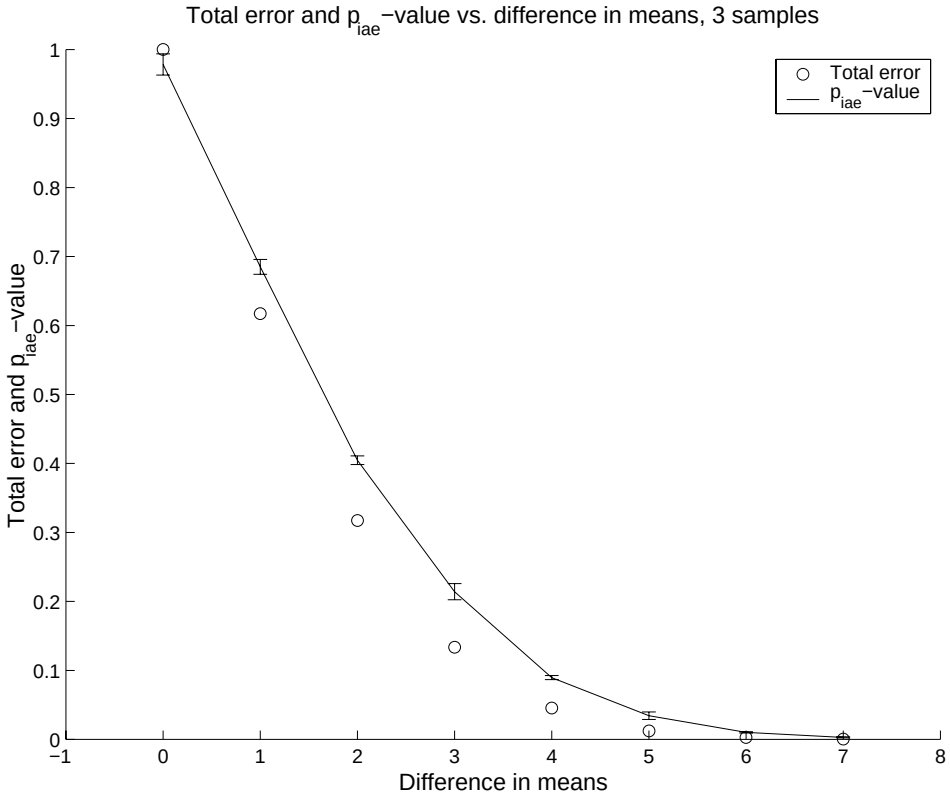

Figure 3.5: True t-test total error and nonparametric p_{iae}-value (mean of three samples) for two normal distributions $N(\mu_1, 1)$ and $N(\mu_2, 1)$ where $\mu_1 = 0$ and $\mu_2 = 0, 1, 2, \ldots, 7$. The nonparametric probability density function estimates were generated from samples of 1000 points from each normal distribution. Table 3.3 presents the numeric values for these data.

Table 3.3: True t-test total error and nonparametric p_{iae}-value for comparing two normal distributions $N(\mu_1, 1)$ and $N(\mu_2, 1)$ where $\mu_1 = 0$ and $\mu_2 = 0, 1, 2, \ldots, 7$. The p_{iae}-value for each pair of means is the average of three trials, and the nonparametric probability density function estimates were generated from samples of 1000 points from each normal distribution.

Difference in means of normal distributions	t-test total error	Nonparametric p_{iae}-value mean (std. dev.)
0.0000	1.0000	0.9730 (0.0102)
1.0000	0.6171	0.6705 (0.0173)
2.0000	0.3173	0.3916 (0.0153)
3.0000	0.1336	0.2095 (0.0081)
4.0000	0.0455	0.0882 (0.0024)
5.0000	0.0124	0.0353 (0.0025)
6.0000	0.0027	0.0096 (0.0008)
7.0000	0.0005	0.0028 (0.0005)

are appropriate for its use as a test statistic, a critical p_{iae}-value must now be determined, generating a decision rule, before the statistic may be used in practice for comparing distributions. To guide the selection of a critical p_{iae}-value a simulation of a real sampling scenario was performed using the standard normal distribution $N(0, 1)$. The simulation consisted of repeating the following three steps 1000 times to obtain an approximation to the distribution of the p_{iae}-value under the null hypothesis of identical distributions for two random samples drawn from a standard normal distribution.

1. Draw a random integer n between 20 and 50 for a sample size.

2. Generate two independent random samples of size n from the standard normal distribution $N(0, 1)$.

3. Compute the p_{iae}-value for the two samples using Equation 3.1 and compute the p-value for the two-sided t-test.

The p_{iae}-value, as previously mentioned, approximates the total error rate for each

sample in this scenario; an error rate near one (1) indicating that the distributions of the two samples are indistinguishable. The two-sided t-test is being used as a goodness of fit test, which, though an unorthodox interpretation, is still consistent with the assumptions and definition of the test. The necessary assumptions for the use of the t-test are: the two samples are from normal distributions with equal variances, having possibly different means. If the means are equal, the distributions are identical, if the means are not equal, the distributions are different, hence the goodness of fit interpretation of the t-test.

A histogram and an empirical cumulative distribution function for the p_{iae}-values obtained from the simulation are presented in Figure 3.6. The distribution of p_{iae}-values is clearly not symmetric and is strongly skewed to the left, with the bulk of the p_{iae}-values appearing above, or to the right of, a p_{iae}-value equal to 0.75. Contrast the distribution of the p_{iae}-values with that of the p-values for the two-sided t-test using the same pairs of samples, given by the histogram and empirical cumulative distribution function in Figure 3.7. The distribution of the two-sided t-test p-values is effectively uniform between zero and one-half, and as such, it cannot provide any insight into the selection of an appropriate critical p-value without some additional knowledge or the arbitrary selection of a small number. This clearly identifies one of the problems of using the t-test for comparing the actual and simulated stands.

The uniform distribution of the p-values for the t-test is to be expected, given the standard use of the p-value and α-level in the classical two-sided t-test hypothesis testing framework [1]. A one-sided t-test would have generated p-values that were uniformly distributed between zero and one. The distinction between a one-sided or two-sided formulation of the t-test is necessary because it uses point estimates to determine the similarity or difference of distributions. The p_{iae}-value, being derived from the data distributions themselves, does not have a one-sided or two-sided formulation; two distributions are either similar or they are different, and this is clearly indicated by the p_{iae}-value.

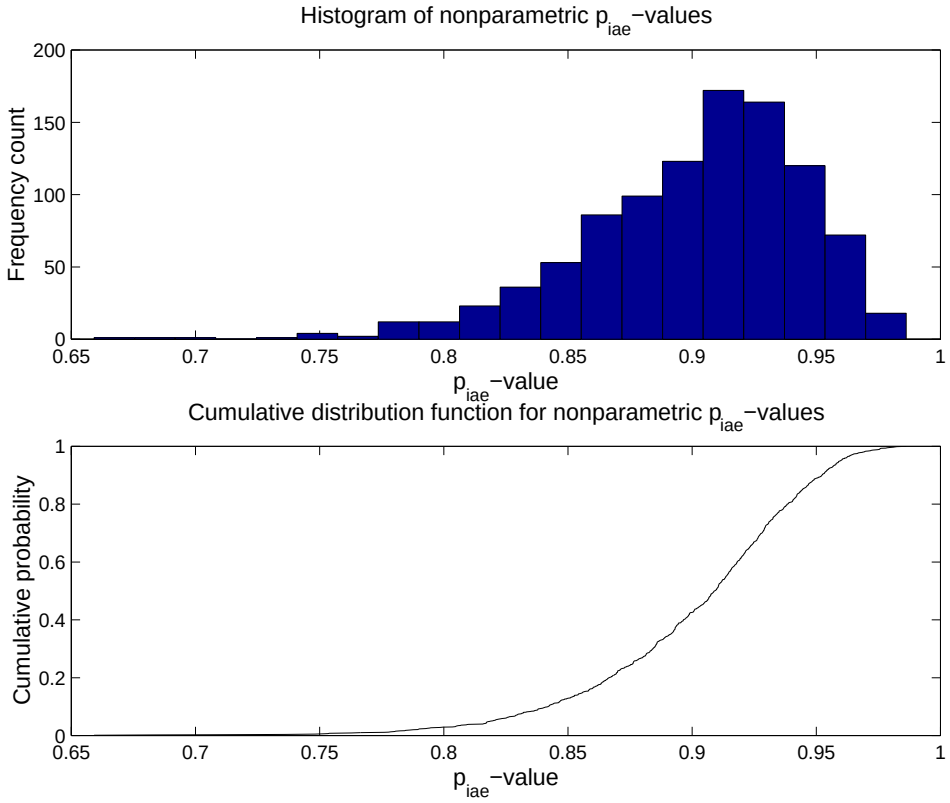

Figure 3.6: Histogram (top) and empirical distribution function (bottom) for 1000 p_{iae}-values. The p_{iae}-values were generated by choosing a random sample size n in the range of 20 to 50 and then drawing a pair of random samples of size n from a standard normal distribution $N(0, 1)$. A p_{iae}-value was then computed for each pair of samples.

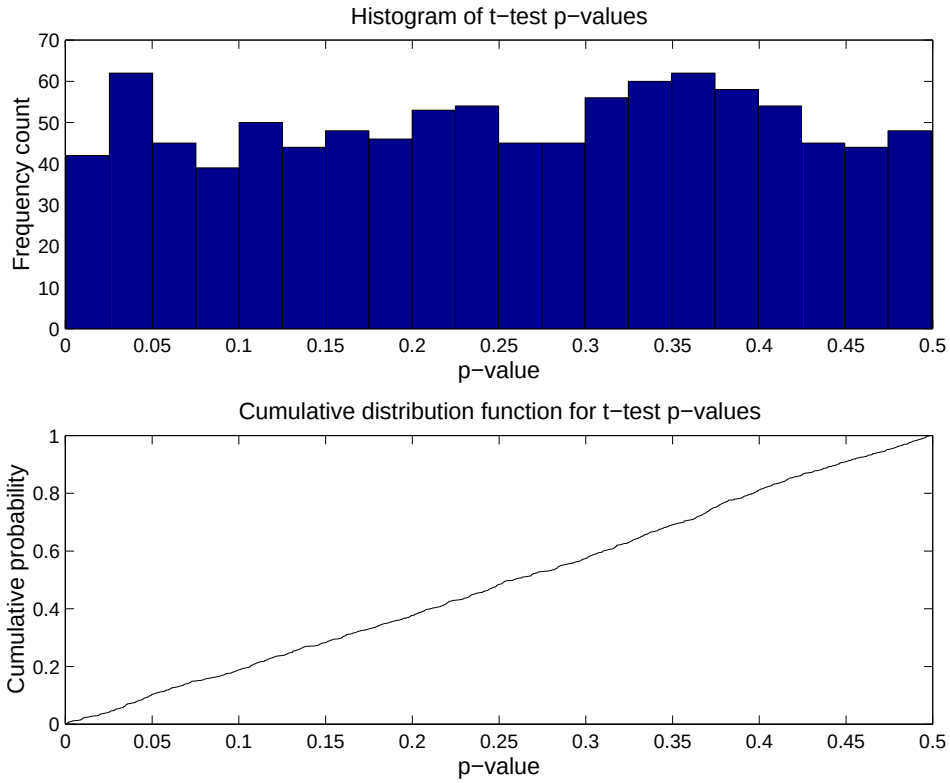

Figure 3.7: Histogram (top) and empirical distribution function (bottom) for 1000 t-test p-values. The p-values were generated from the same pairs of samples used for the p_{iae}-values in Figure 3.6. The p-values were computed using a two sided t-test.

Figure 3.8 and Figure 3.9 present the nonparametric probability density function estimates, sample size n, p-value, and p_{iae}-value for two very similar random samples and two somewhat different random samples, respectively, from the sampling simulation. The agreement between the two distributions is approximately 98%, a p_{iae}-value of 0.98, but the p-value indicates a 94% agreement, a p-value of 0.47 for Figure 3.8. The distributions in Figure 3.9 are obviously different, with an agreement of approximately 74%, a p_{iae}-value of 0.74, but the t-test p-value indicates that the means, and hence the distributions, are highly significantly different, a p-value of 0.004. This demonstrates another common problem with the t-test, the spurious rejection of the null hypothesis.

Given the *a priori* knowledge that the two samples used to obtain each p_{iae}-value or t-test p-value were drawn from the same underlying distribution, the standard normal distribution $N(0, 1)$, the p_{iae}-value clearly proves to be more effective as an indicator of the similarity of the distributions. The p_{iae}-value also provides a strong indication of a similarity threshold value, whereas the t-test p-value does not. This is not surprising, since the p_{iae}-value uses more of the information contained in the samples than the t-test, so it should perform better in this regard. It is for this particular reason that the p_{iae}-value has been chosen as the primary test statistic, rather than the t-test and its p-value, for comparisons of the actual and simulated stands for the validation of the tree list generation database methodology.

An appropriate critical p_{iae}-value may be chosen by considering the theoretical interpretation of the p_{iae}-value guided by the results of the sampling simulation. Recall that the p_{iae}-value gives a direct indication of the proportion of probability mass that two continuous distributions have in common. A p_{iae}-value of one indicating that the two distributions have all of their probability mass in common, i.e., they are identical, and a p_{iae}-value of zero indicating that the two distributions have none of their probability mass in common, i.e., they are completely different. The sampling simulation indicates that for two samples drawn from a standard normal distribution,

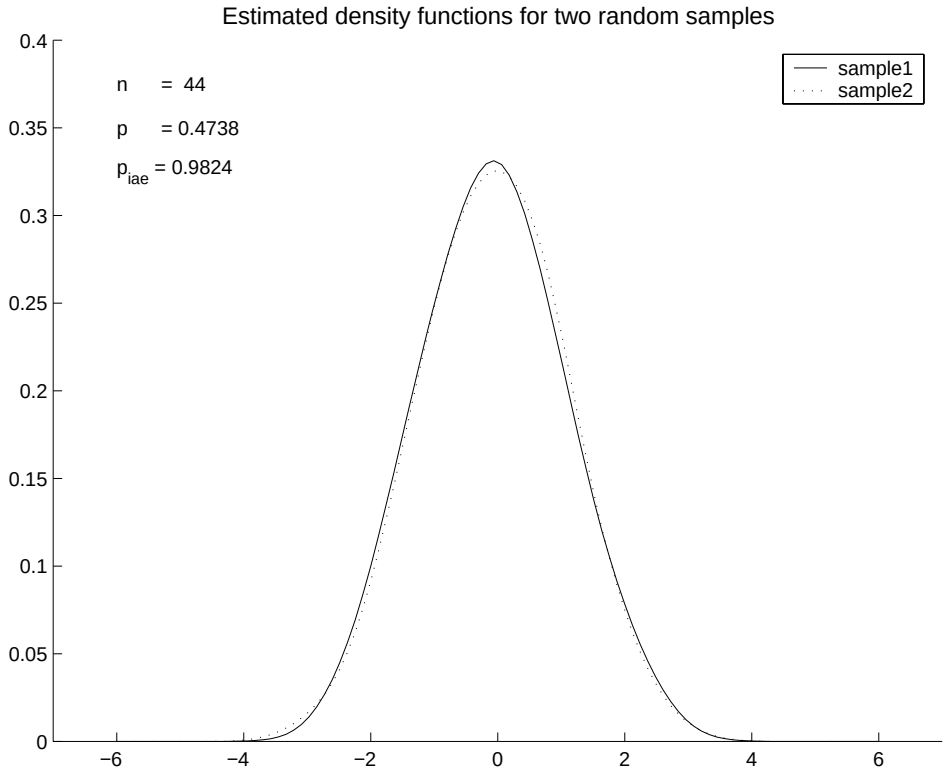

Figure 3.8: Nonparametric density function estimates for two samples of size 44 from a standard normal distribution $N(0, 1)$. The two density functions are almost identical, with a p_{iae}-value of approximately 0.98. The t-test p-value of 0.47 for the difference in means in this example, indicates that the means are not significantly different, but gives a slightly poorer indication of the actual similarity of the distributions.

most of the p_{iae}-values are greater than a p_{iae}-value of 0.75, or 75% similarity between distributions. So, a p_{iae}-value of 0.75 seems to be a reasonable critical value to use in practice; p_{iae}-values ≥ 0.75 indicating similarity between distributions, and p_{iae}-values < 0.75 indicating lack of similarity between distributions. The critical p_{iae}-value of 0.75 will, in general, be conservative for distributions that are multimodal or that are not strongly unimodal like the normal distribution; more variation in the distribution of probability mass for a particular pair of distributions will lead to a lower critical p_{iae}-value when comparing distributions.

The p_{iae}-value is the test statistic used to computee the empirical misclassification

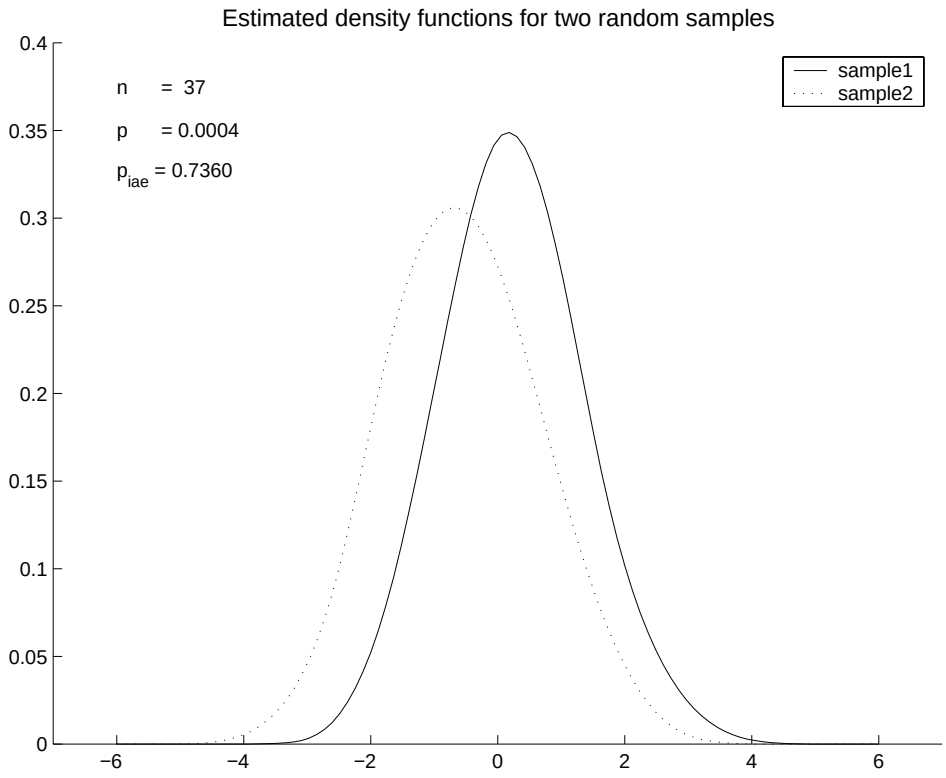

Figure 3.9: Nonparametric density function estimates for two samples of size 37 from a standard normal distribution $N(0,1)$. The two density functions are obviously different, but the p_{iae}-value of 0.74 indicates that they have 75% of their probability mass in common. The t-test p-value of 0.004 for the difference in means in this example, indicates that the means, and hence distributions, are highly significantly different.

rates from the comparisons between actual stands and simulated stands in the validation of the tree list generation database using the critical p_{iae}-value of 0.75. The continuous and categorical p_{iae}-value formulae, Equation 3.1 and Equation 3.2, are used to compute the p_{iae}-values. The histogram based formula, Equation 3.3 was provided for completeness, but it could have been used instead of the continuous formula, Equation 3.1. Comparisons involving the actual and simulated DBH and height distributions, which are nominally from continuous distributions, Equation 3.1 and the nonparametric probability density function estimation and numerical integration procedures described above are used. Comparisons involving the species compositions of the actual and simulated stands, use Equation 3.2 with the proportion of each species in either the actual or simulated stand.

For each comparison between an actual and a simulated stand, a p_{iae}-value is computed for each of the DBH and height distributions, and species composition. For each distribution, a p_{iae}-value greater than or equal to the critical value of 0.75 indicates that their distributions are not distinguishable, i.e are the same, and a p_{iae}-value less than the critical value of 0.75 indicates that their distributions are different. If a p_{iae}-value is less than the critical value of 0.75 for a particular attribute and pair of stands, a misclassification error for that attribute has occurred; the simulated stand was generated to mimic the characteristics of the actual stand. Empirical misclassification rates for each attribute, and a total misclassification rate, provide the primary form of validation for the tree list generation database methodology. The total misclassification rate is based on the logical *or*ing of the three individual classification outcomes. If at least one of DBH, height, or species composition is misclassified for a particular pair of actual and simulated stands, this is considered as a misclassified stand for the computation of the total misclassification rate. The total misclassification rate, therefore, provides an upper bound on the true total misclassification rate.

There are four tree list generation database testing scenarios. Each of the four

testing scenarios is applied to the two treatments in the tree list generation database `tgdb1r00` to obtain a set of misclassification rates. The testing scenarios correspond to the four possible stand generation modes that may be specified in a stand description file used with `TGRAND` when generating a simulated stand. The four possible stand generation modes are listed in Table 3.4, and they are determined by the values of the `RANDOM_MF_JITTER` and `RANDOM_MF_PURE` keywords specified in a stand description file, Section 2.11.2. When presenting the goodness of fit testing results, the misclassification rates, only figures for scenario RMF are presented. The figures for the other testing scenarios are similar. Tables summarizing the primary goodness of fit testing results for all four testing scenarios are presented for comparison with the results of scenario RMF. The RMF testing scenario is the default stand generation mode and also provides the worst case scenario for stands generated using `TGRAND`.

Table 3.4: The four simulated tree list generation modes for the tree list generation database program `TGRAND`. Jittering heights adds or subtracts a small random amount to or from each height value in an attempt to represent the natural variability of tree heights. The 100% pure stand mode forces generated stands to be 100% of the dominant species if they have a stand type of `PURE_DF` or `PURE_WH`. For details on how to set these properties for a tree list see Section 2.11.2.

Scenario	100% pure stands	Jitter heights
RMF (default)	No	No
RMFJ	No	Yes
RMFP	Yes	No
RMFJP	Yes	Yes

Results are summarized in three different ways for each treatment. First, misclassification rates are presented for each testing scenario in DBH, height, species composition, and a total misclassification rate. These error rates test the within stand agreement of each actual and simulated stand and represent the bulk of the tree list generation database validation results. Histograms of the DBH, height, and species composition p_{iae}-values are also presented, with the nominal p_{iae} cutoff, or

critical value, of 0.75, 75% probability mass overlap indicated. These figures provide a graphical sense of the misclassification rates for different p_{iae} cutoff values. Second, scatter plots of simulated and actual QMD and mean height are presented with linear regression coefficients and r^2 values. Third, the distributions of QMD and mean height across all actual and simulated stands are compared. These two comparisons provide an indication of how well these stand attributes are reproduced by the simulated stands. The comparisons test the ability of the tree list generation database to reproduce the appropriate regional properties across all of the stands in a database, or the between stand compatibility of the actual and simulated stands.

The tree list generation database goodness of fit testing results are presented in the next two sections. First are the results for untreated stands. Second are the results for thinned stands. See Section 3.4.1 for some additional information on the interpretation of the results, particularly the relevance of the critical p_{iae}-value of 0.75 to forestry applications; this critical p_{iae}-value may be quite conservative. Also see Section 3.4.2 for an example of using the tree list generation database with forest growth and yield models or forest simulation systems.

3.3.3 Goodness of fit testing results for untreated stands

The results are for the tree list generation database `tgdb1r00` which used the stand index parameters

```
TREATMENT_TYPE
SITE_INDEX_50
STAND_TOTAL_AGE
STAND_ORIGIN
QMD
STAND_DENSITY
STAND_TYPE
```

for the stand classification and the nearest neighbor stand selection procedures. See Section 2.11.6 for definitions of the index parameters, and Section 3.1.1 and Section 3.1.2 for the stand classification and stand selection procedures, respectively.

The primary results provide empirical misclassification rates for DBH and height distributions, species composition, and a total misclassification rate derived from the individual DBH height, and species misclassification rates. The remaining results compare the stand level attributes of QMD and mean height.

The empirical misclassification rate results were generated by first creating a simulated stand to mimic each of the untreated stand measurements used to populaste the tree list generation database `tgdb1r00`. Each pair of actual and simulated stands were then compared by computing a set of p_{iae}-values for the DBH distribution, the height distribution, and the species composition for each pair of actual and simulated stand measurements. Empirical misclassification rates for each distribution were then generated by performing hypothesis tests using the p_{iae}-values for the three individual distributions, and the critical p_{iae}-value of 0.75. Distributions which had p_{iae}-values ≥ 0.75 were considered to be similar, and distributions which had p_{iae}-values < 0.75 were considered to be different, indicating a misclassification. An overall, or total, misclassification rate was also computed from the individual distribution misclassification rates. This process was repeated for each of the four tree list generation database testing scenarios in Table 3.4.

The empirical misclassification rate results for the DBH distributions, the height distributions, and the species composition for untreated stands and testing scenario RMF are presented in Figure 3.10, Figure 3.11, and Figure 3.12, respectively. The figures clearly show that the bulk of the actual stand *vs.* simulated stand comparisons had p_{iae}-values that were greater than or equal to the critical p_{iae}-value of 0.75, indicating that the majority of simulated untreated stands were indeed similar to their actual counterparts. The misclassification rates for testing scenario RMF were 2.59% for DBH distributions, 13.21% for height distributions, 8.25% for species composition, and 19.72% overall. The larger misclassification rate for height distributions is most likely due to the reduction in the natural variation in tree heights caused by the estimation of tree heights from height-diameter relationships. Table 3.5 summarizes

the misclassification rate results for the other three testing scenarios, including the results for scenario RMF for comparison.

The results for testing scenario RMF represent the worst case scenario for the generated stands, and also represent the default mode of operation for the tree list generation database program `TGRAND`. As can be seen, the worst case performance is quite good, having an overall correct classification rate in excess of 80%, demonstrating that there is very good agreement between the actual and simulated untreated stands at the level of individual stand measurements. The results for the other three testing scenarios prove to be slightly better than the 80% correct classification rate of testing scenario RMF, as expected, with the two scenarios having jittered tree heights both performing better than their nonjittered counterparts.

Table 3.5: Misclassification rates in DBH, height, and species for untreated stands. Values are the proportion of stands misclassified for each variable. The total column gives the proportion of stands which were misclassified in at least one of DBH, height, or species, and gives an upper bound on the actual misclassification rate.

Scenario	DBH	Height	Species	Total
RMF	0.0259	0.1321	0.0825	0.1972
RMFJ	0.0261	0.1244	0.0781	0.1851
RMFP	0.0255	0.1298	0.0793	0.1929
RMFJP	0.0257	0.1221	0.0731	0.1801

Results for the comparison of the aggregate properties of the actual and simulated untreated stands was performed using QMD and average tree height as the stand level attributes. The comparison consisted of performing a simple linear regression of the simulated QMD and mean height values against the actual QMD and mean height values, respectively. For this analysis, an intercept of zero (0) and a slope of one (1) indicate exact agreement between the simulated and actual QMD or mean height values.

The QMD data used in the simple linear regression analysis for testing scenario RMF are given in Figure 3.13. A cursory examination of the figure clearly indicates

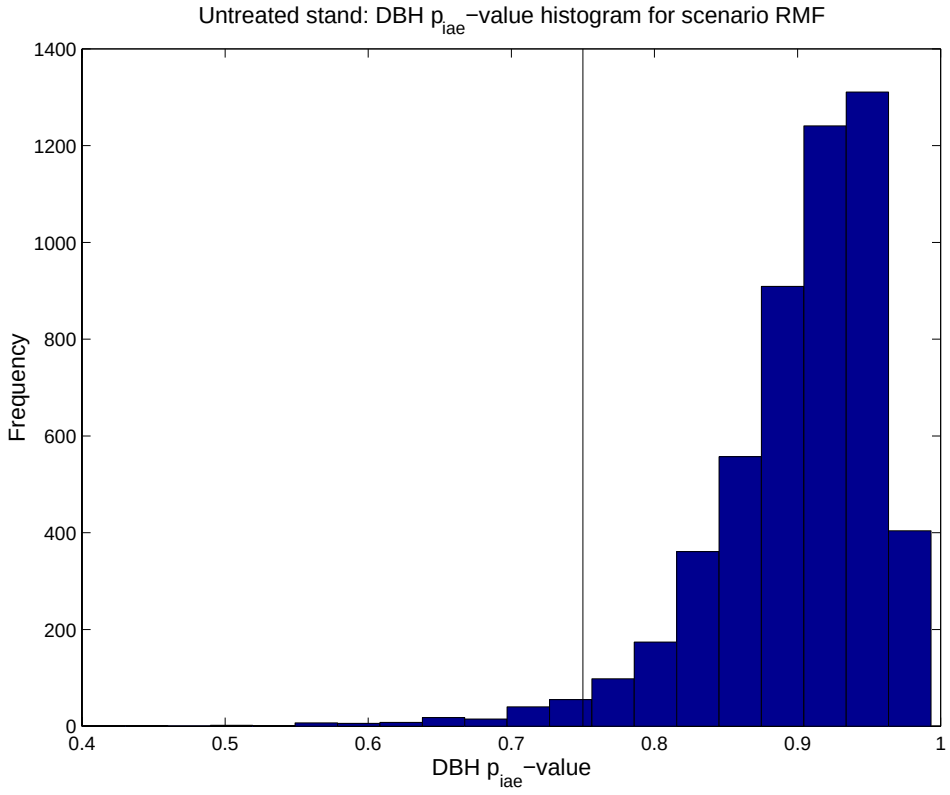

Figure 3.10: Histogram of p_{iae}-values for tree DBH in untreated stands for scenario RMF. The vertical line at 0.75 represents the boundary between the similar (p_{iae}-value$\geq$ 0.75) and different (p_{iae}-value$<$ 0.75) regions for the p_{iae}-values. The cutoff value of 0.75 indicates a 75% agreement between the actual and simulated DBH distributions. The misclassification rate is 2.59%. See Table 3.5 for error rates for the other scenarios.

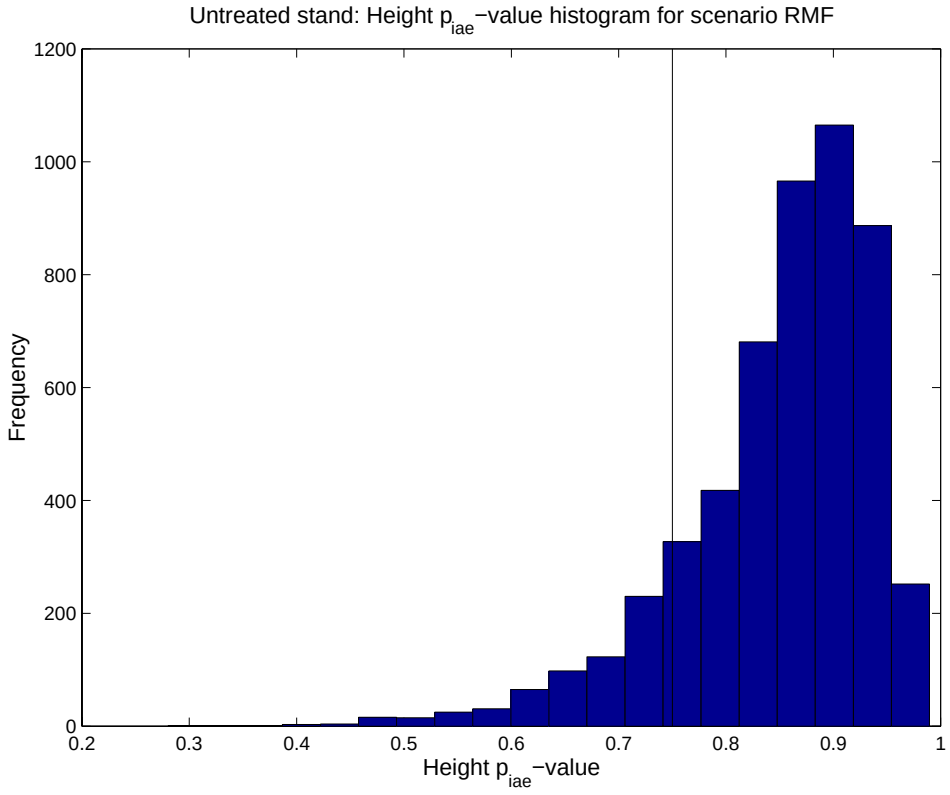

Figure 3.11: Histogram of p_{iae}-values for tree height in untreated stands for scenario RMF. The vertical line at 0.75 represents the boundary between the similar (p_{iae}-value$\geq$ 0.75) and different (p_{iae}-value$<$ 0.75) regions for the p_{iae}-values. The cutoff value of 0.75 indicates a 75% agreement between the actual and simulated height distributions. The misclassification rate is 13.21%. See Table 3.5 for error rates for the other scenarios.

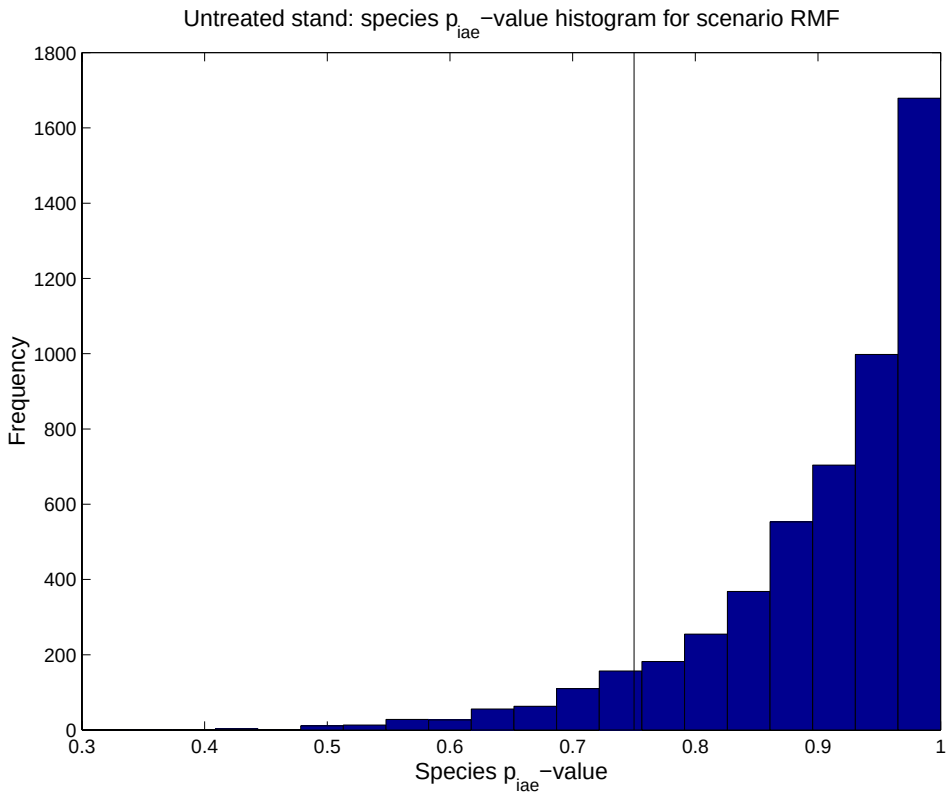

Figure 3.12: Histogram of p_{iae}-values for species composition in untreated stands for scenario RMF. The vertical line at 0.75 represents the boundary between the similar (p_{iae}-value$\geq$ 0.75) and different (p_{iae}-value$<$ 0.75) regions for the p_{iae}-values. The misclassification rate is 8.25%. See Table 3.5 for error rates for the other scenarios.

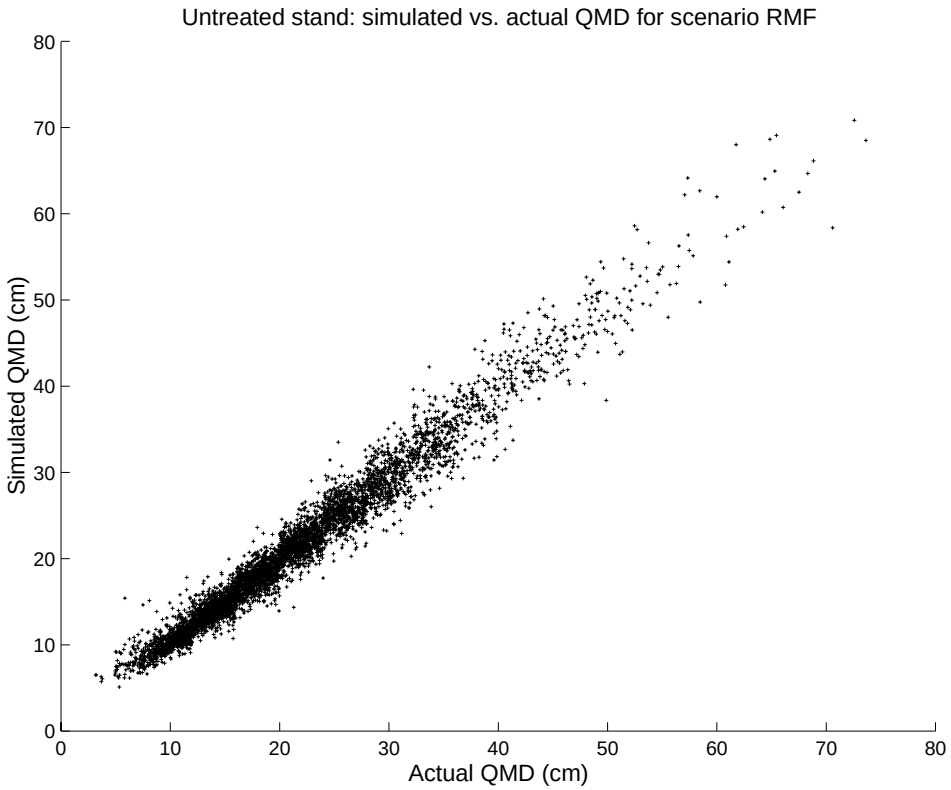

Figure 3.13: Plot of simulated *vs.* actual QMD values (cm) from untreated stands for scenario RMF. Linear regression coefficients for the model $y = a + bx$ applied to these data yields the model $y = 0.7766 + 0.9578x$ with $r^2 = 0.9558$. See Table 3.6 for linear regression coefficients and r^2 values for the other scenarios.

a strong linear relationship between the actual and simulated values for this stand attribute. The QMD intercept of 0.7766, the slope of 0.9578 and the r^2 value of 0.9558 support the observation of strong linearity. Table 3.6 presents the linear regression coefficients and r^2 values for the other three testing scenarios, as well as the results for scenario RMF for comparison. These results indicate that there is very good agreement for QMD measurements between the actual and simulated untreated stands for all four testing scenarios.

The mean tree height data used in the simple linear regression analysis for testing scenario RMF are given in Figure 3.14. Again, a cursory examination of the figure clearly indicates a strong linear relationship between the actual and simulated values

Table 3.6: Linear regression coefficients and r^2 values for the model $y = a+bx$ applied to the simulated (y) and actual (x) QMD values (cm) for untreated stands. A perfect fit would give values $a = 0$ and $b = 1$.

Scenario	Intercept (a)	Slope (b)	r^2
RMF	0.7766	0.9578	0.9558
RMFJ	0.8073	0.9547	0.9534
RMFP	0.7223	0.9648	0.9564
RMFJP	0.7743	0.9603	0.9535

for this stand attribute. The mean height intercept of 1.1419, the slope of 0.9403 and the r^2 value of 0.8720 support the observation of strong linearity. Table 3.7 presents the linear regression coefficients and r^2 values for the other three testing scenarios, as well as the results for scenario RMF for comparison. These results indicate that there is very good agreement for mean height measurements between the actual and simulated untreated stands for all four testing scenarios.

Table 3.7: Linear regression coefficients and r^2 values for the model $y = a+bx$ applied to the simulated (y) and actual (x) mean height values (m) for untreated stands. A perfect fit would give values $a = 0$ and $b = 1$.

Scenario	Intercept (a)	Slope (b)	r^2
RMF	1.1419	0.9403	0.8720
RMFJ	1.1248	0.9400	0.8686
RMFP	1.0018	0.9535	0.8721
RMFJP	0.9919	0.9529	0.8693

Finally, the distributions of the aggregate stand level attributes QMD and mean tree height for untreated stands were compared. The distributions for these stand attributes were compared by computing a p_{iae}-value for each pair of actual and simulated QMD and mean height distributions. Figure 3.15 and Figure 3.16 present the nonparametric probability density estimates for the actual and simulated QMD and mean height distributions for testing scenario RMF. Visually, the actual and simulated distributions are quite similar for both QMD and mean height. The p_{iae}-values

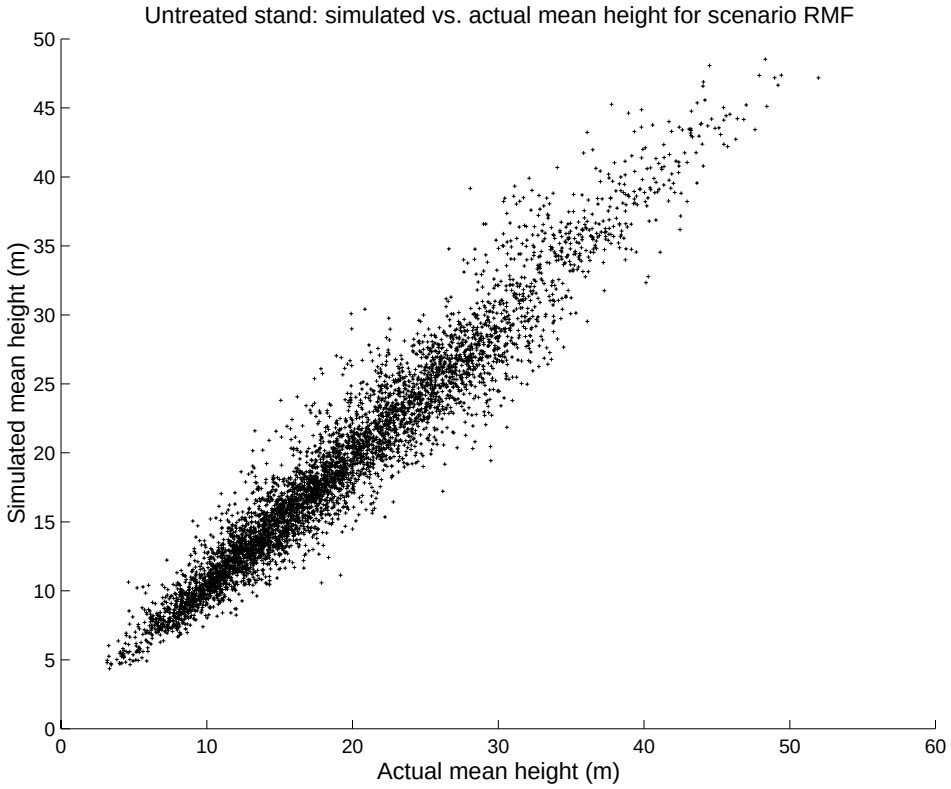

Figure 3.14: Plot of simulated *vs.* actual mean height values (m) from untreated stands for scenario RMF. Linear regression coefficients for the model $y = a + bx$ applied to these data yields the model $y = 1.1419 + 0.9403x$ with r^2= 0.8720. See Table 3.7 for linear regression coefficients and r^2 values for the other scenarios.

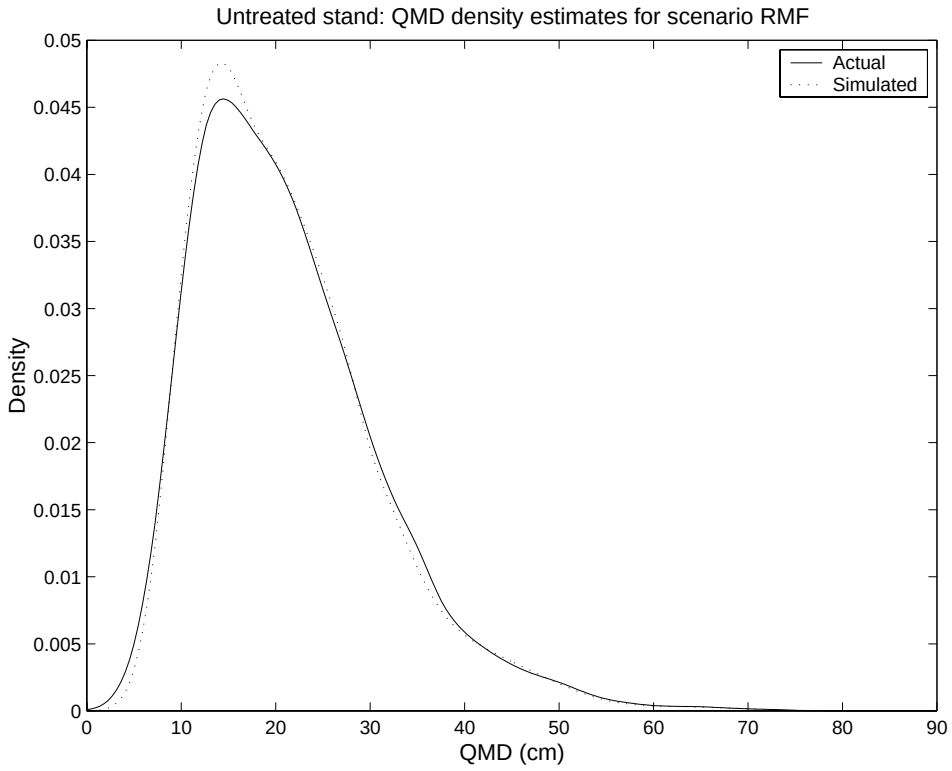

Figure 3.15: Nonparametric probability density function estimates for the actual and simulated QMD distributions for all untreated stands and scenario RMF. The p_{iae}-value for the actual *vs.* simulated QMD distributions is 0.9778. See Table 3.8 for the overall QMD distribution p_{iae}-values for the other scenarios.

for the QMD and mean height distributions for scenario RMF are 0.9778 and 0.9779, respectively, indicating that the actual and simulated QMD and mean height distributions are almost identical, having approximately 98% of their probability mass in common. Table 3.8 presents the results for the other three testing scenarios as well as the results for scenario RMF for comparison. These results reinforce the results from the simple linear regressions, and provide another indication of how well the tree list generation database methodology performs. Notice, in particular, that the distribution of mean height for the simulated stands reproduces the kink in the actual mean height distribution near the mean height of 15 m (49 ft).

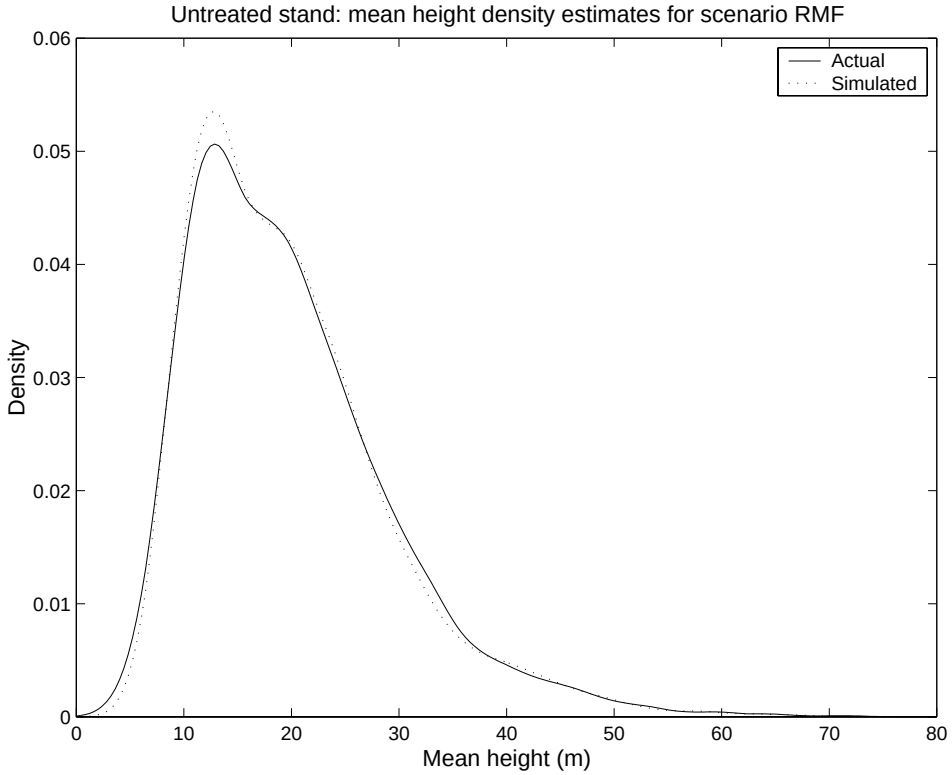

Figure 3.16: Nonparametric probability density function estimates for the actual and simulated mean height distributions for all untreated stands and scenario RMF. The p_{iae}-value for the actual *vs.* simulated mean height distributions is 0.9779. See Table 3.8 for the overall mean height distribution p_{iae}-values for the other scenarios.

Table 3.8: Overall QMD and mean height distribution p_{iae}-values for actual and simulated untreated stands for all four scenarios. See Figure 3.15 and Figure 3.16.

Scenario	QMD p_{iae}-value	Mean height p_{iae}-value
RMF	0.9778	0.9779
RMFJ	0.9762	0.9741
RMFP	0.9813	0.9804
RMFJP	0.9799	0.9790

3.3.4 Goodness of fit testing results for thinned stands

The results are for the tree list generation database `tgdb1r00` which used the index parameters

```
TREATMENT_TYPE
SITE_INDEX_50
STAND_TOTAL_AGE
STAND_ORIGIN
QMD
STAND_DENSITY
STAND_TYPE
NUMBER_OF_THINNINGS
PRE_THIN_DENSITY
PRE_THIN_BASAL_AREA
PCT_BASAL_AREA_REMOVED
YEARS_SINCE_TREATMENT
```

for the stand classification and the nearest neighbor stand selection. See Section 2.11.6 for definitions of the index parameters, and Section 3.1.1 and Section 3.1.2 for the stand classification and stand selection procedures, respectively. The primary results provide empirical misclassification rates for DBH and height distributions, species composition, and a total misclassification rate derived from the individual DBH height, and species misclassification rates. The remaining results compare the stand level attributes of QMD and mean height.

The empirical misclassification rate results were generated by first creating a simulated stand to mimic each of the thinned stand measurements used to populate the tree list generation database `tgdb1r00`. Each pair of actual and simulated stands were then compared by computing a set of p_{iae}-values for the tree DBH distribution, the tree height distribution, and the species composition. Empirical misclassification rates for each distribution were then generated by performing hypothesis tests using the p_{iae}-values for the three individual distributions, and the critical p_{iae}-value of 0.75. Distributions which had p_{iae}-values ≥ 0.75 were considered to be similar, and distributions which had p_{iae}-values < 0.75 were considered to be different, indicating a misclassification. An overall, or total, misclassification rate was also computed from

the individual distribution misclassification rates. This process was repeated for each of the four tree list generation database testing scenarios in Table 3.4.

The empirical misclassification rate results for the DBH distributions, the height distributions, and the species composition for thinned stands and testing scenario RMF are presented in Figure 3.17, Figure 3.18, and Figure 3.19, respectively. The figures clearly show that the bulk of the actual stand *vs.* simulated stand comparisons had p_{iae}-values that were greater than or equal to the critical p_{iae}-value of 0.75, indicating that the majority of the simulated thinned stands were indeed similar to their actual counterparts. The misclassification rates for testing scenario RMF were 5.68% for DBH distributions, 15.02% for height distributions, 2.75% for species composition, and 18.47% overall. The larger misclassification rate for height distributions is most likely due to the reduction in the natural variation in tree heights caused by the estimation of tree heights from height-diameter relationships, and the further reduction in the height variation caused by the thinning. The higher DBH misclassification rate, compared to that of untreated stands, 2.59%, is also most likely due to smaller variation in the DBH measurements caused by the thinning. The lower misclassification rate for species composition is also directly attributable to the thinning: undesirable tree species would have been removed. Table 3.9 summarizes the misclassification rate results for the other three testing scenarios, including the results for scenario RMF for comparison.

The results for testing scenario RMF represent the worst case scenario for the generated stands, and also represent the default mode of operation for the tree list generation database program `TGRAND`. As can be seen, the worst case performance is quite good, with an overall correct classification rate in excess of 81%, demonstrating that there is very good agreement between the actual and simulated stands at the level of individual stand measurements for the thinned stands. The results for two of the other three testing scenarios prove to be slightly better than the 81% correct classification rate of testing scenario RMF, as expected,but the results for the

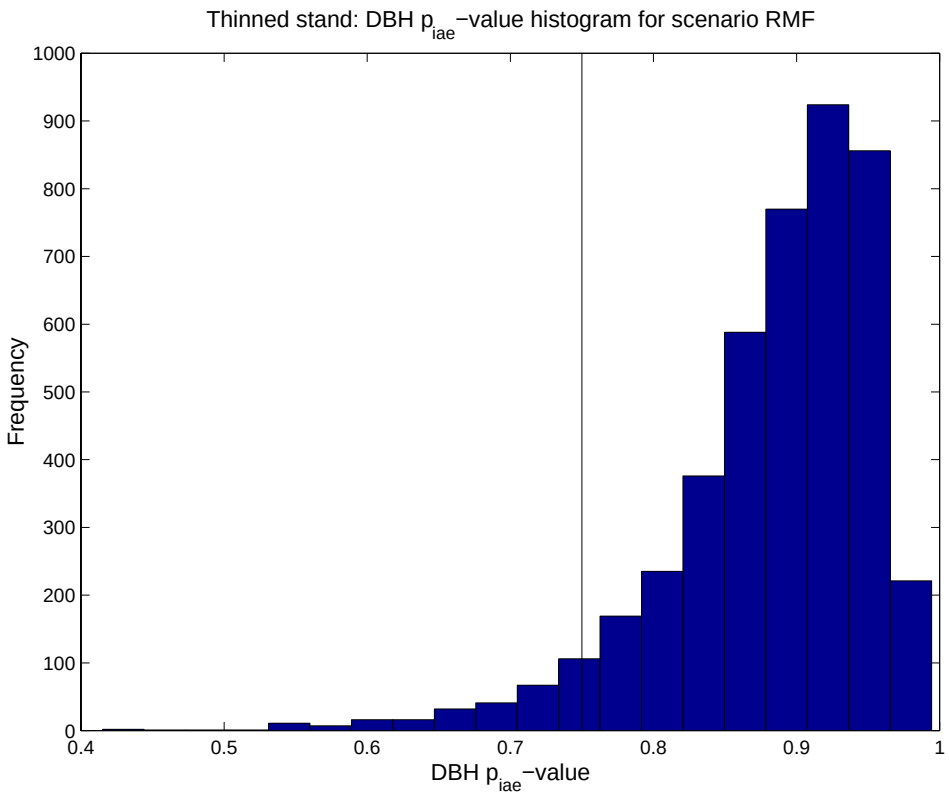

Figure 3.17: Histogram of p_{iae}-values for tree DBH in thinned stands for scenario RMF. The vertical line at 0.75 represents the boundary between the similar (p_{iae}-value$\geq$ 0.75) and different (p_{iae}-value$<$ 0.75) regions for the p_{iae}-values. The cutoff value of 0.75 indicates a 75% agreement between the actual and simulated DBH distributions. The misclassification rate is 5.68%. See Table 3.9 for error rates for the other scenarios.

remaining scenario, scenario RMFJ, are nearly identical to the results for scenario RMF. The two testing scenarios having jittered heights performed better than their nonjittered counterparts, but scenario RMFJ slightly outperformed scenario RMFJP which should have been the best case scenario. The difference is small, 0.45%, and is most likely a result of the *luck of the draw* when generating simulated trees

The results for the comparison of the aggregate properties of the actual and simulated thinned stands was performed using QMD and average tree height as the stand level attributes. The comparison consisted of performing a simple linear regression

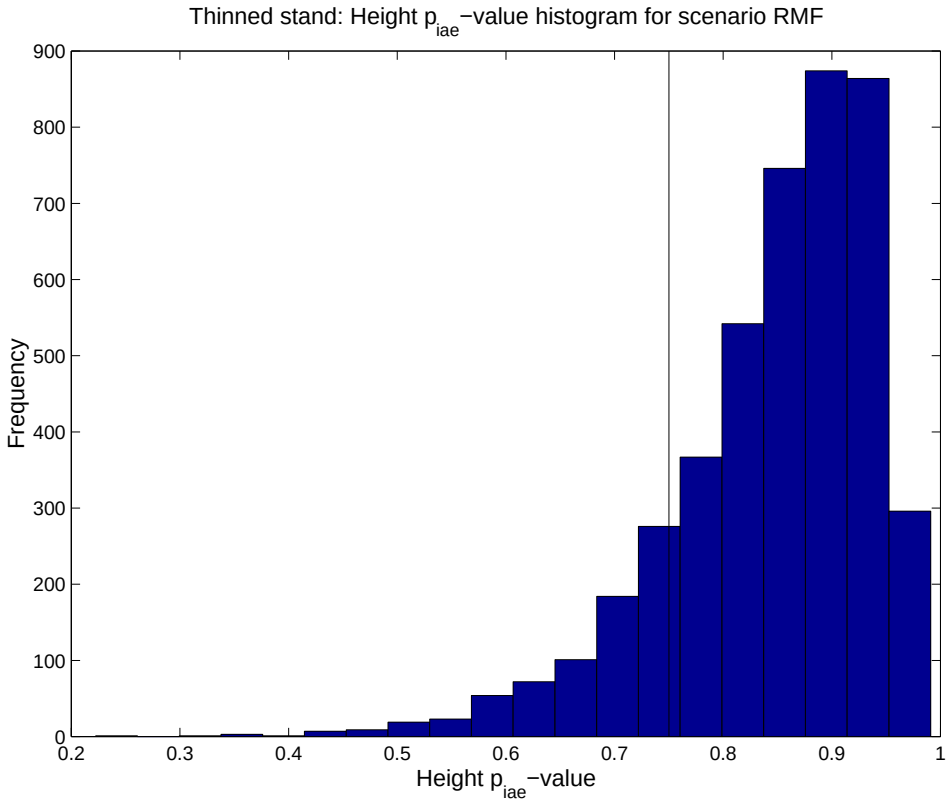

Figure 3.18: Histogram of p_{iae}-values for tree height in thinned stands for scenario RMF. The vertical line at 0.75 represents the boundary between the similar (p_{iae}-value$\geq$ 0.75) and different (p_{iae}-value$<$ 0.75) regions for the p_{iae}-values. The cutoff value of 0.75 indicates a 75% agreement between the actual and simulated height distributions. The misclassification rate is 15.02%. See Table 3.9 for error rates for the other scenarios.

Table 3.9: Misclassification rates in DBH, height, and species for thinned stands. Values are the proportion of stands misclassified for each variable. The total column gives the proportion of stands which were misclassified in at least one of DBH, height, or species, and gives an upper bound on the actual misclassification rate.

Scenario	DBH	Height	Species	Total
RMF	0.0568	0.1502	0.0275	0.1800
RMFJ	0.0595	0.1502	0.0225	0.1730
RMFP	0.0588	0.1529	0.0230	0.1804
RMFJP	0.0590	0.1509	0.0223	0.1775

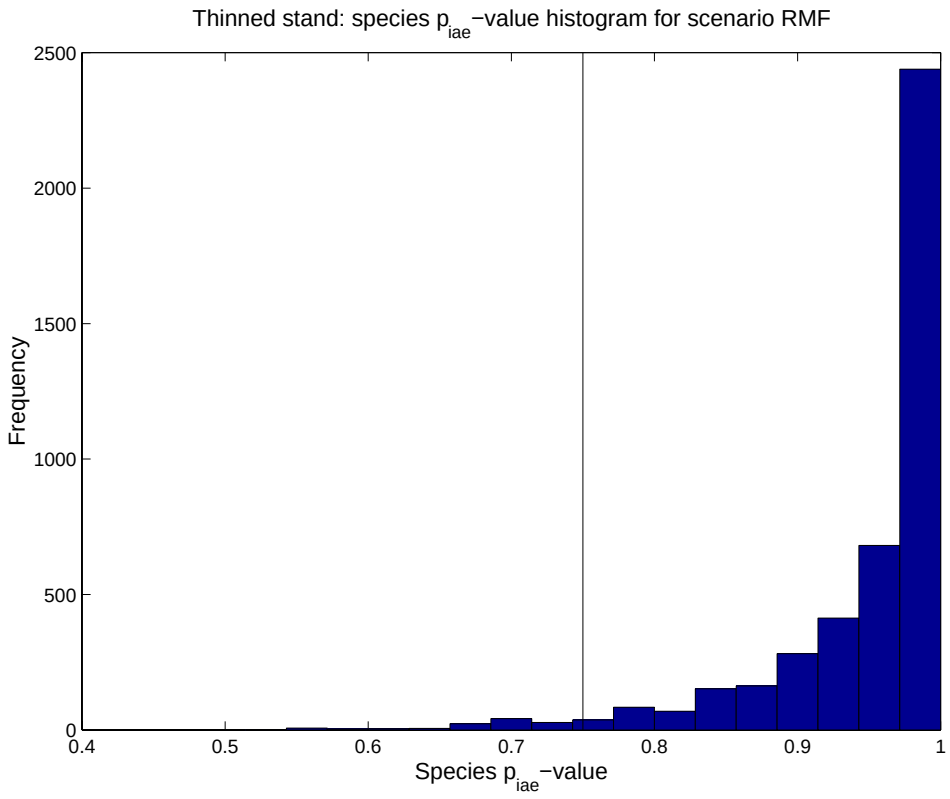

Figure 3.19: Histogram of p_{iae}-values for species composition in thinned stands for scenario RMF. The vertical line at 0.75 represents the boundary between the similar (p_{iae}-value$\geq$ 0.75) and different (p_{iae}-value$<$ 0.75) regions for the p_{iae}-values. The misclassification rate is 2.75%. See Table 3.9 for error rates for the other scenarios.

of the simulated QMD and mean height values against the actual QMD and mean height values, respectively. For this analysis, an intercept of zero (0) and a slope of one (1) indicate exact agreement between the simulated and actual QMD or mean height values.

The QMD data used in the simple linear regression analysis for testing scenario RMF are given in Figure 3.20. A cursory examination of the figure clearly indicates a strong linear relationship between the actual and simulated values for this stand attribute. The QMD intercept of 0.6959, the slope of 0.9635 and the r^2 value of 0.9496 support the observation of strong linearity. Table 3.10 presents the linear regression coefficients and r^2 values for the other three testing scenarios, as well as the results for scenario RMF for comparison. These results indicate that there is very good agreement for QMD measurements between the actual and simulated thinned stands for all four testing scenarios.

Table 3.10: Linear regression coefficients and r^2 values for the model $y = a + bx$ applied to the simulated (y) and actual (x) QMD values (cm) for thinned stands. A perfect fit would give values $a = 0$ and $b = 1$.

Scenario	Intercept (a)	Slope (b)	r^2
RMF	0.6959	0.9635	0.9496
RMFJ	0.8613	0.9556	0.9492
RMFP	0.6912	0.9652	0.9526
RMFJP	0.6032	0.9693	0.9492

The mean tree height data used in the simple linear regression analysis for testing scenario RMF are given in Figure 3.21. Again, a cursory examination of the figure clearly indicates a strong linear relationship between the actual and simulated values for this stand attribute. The mean height intercept of 0.4151, the slope of 0.9791 and the r^2 value of 0.8806 support the observation of strong linearity. Table 3.11 presents the linear regression coefficients and r^2 values for the other three testing scenarios, as well as the results for scenario RMF for comparison. These results indicate that

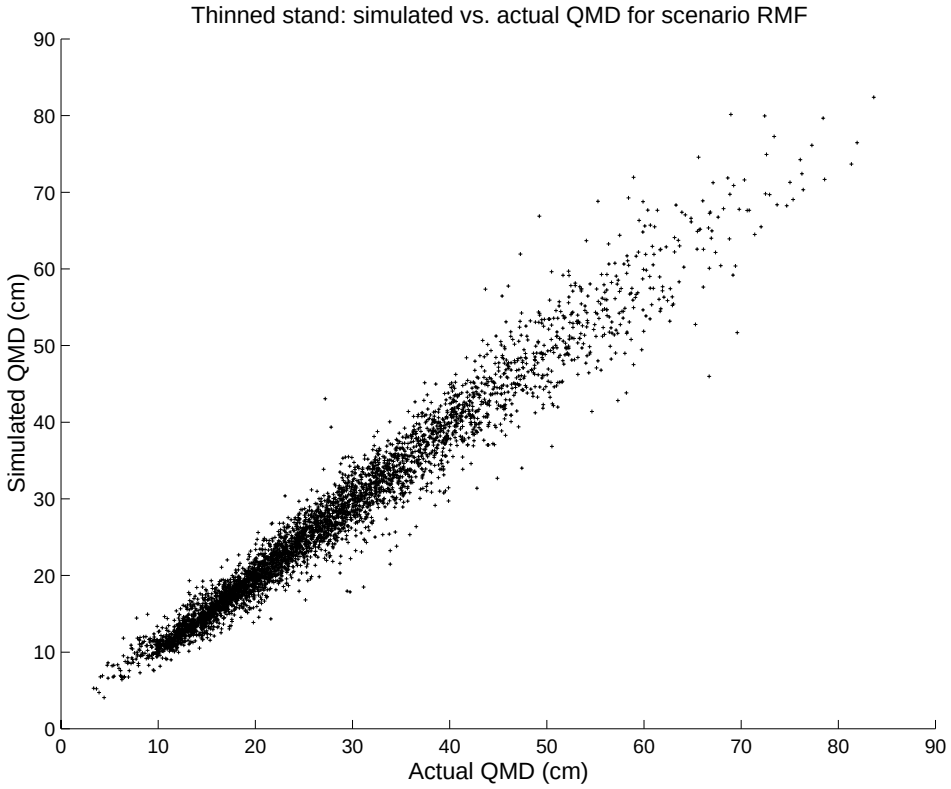

Figure 3.20: Plot of simulated *vs.* actual QMD values (cm) from thinned stands for scenario RMF. Linear regression coefficients for the model $y = a + bx$ applied to these data yields the model $y = 0.6959 + 0.9635x$ with $r^2 = 0.9496$. See Table 3.10 for linear regression coefficients and r^2 values for the other scenarios.

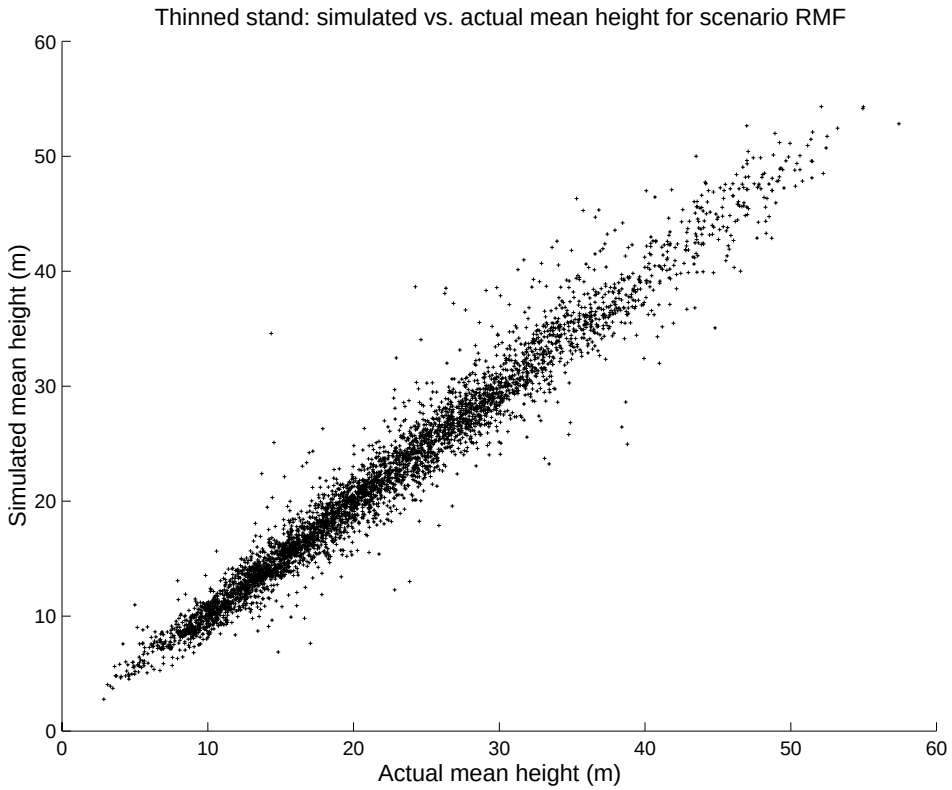

Figure 3.21: Plot of simulated *vs.* actual mean height values (m) from thinned stands for scenario RMF. Linear regression coefficients for the model $y = a + bx$ applied to these data yields the model $y = 0.4151 + 0.9791x$ with r^2= 0.8806. See Table 3.11 for linear regression coefficients and r^2 values for the other scenarios.

there is very good agreement for mean height measurements between the actual and simulated thinned stands for all four testing scenarios.

Table 3.11: Linear regression coefficients and r^2 values for the model $y = a+bx$ applied to the simulated (y) and actual (x) mean height values (m) for thinned stands. A perfect fit would give values $a = 0$ and $b = 1$.

Scenario	Intercept (a)	Slope (b)	r^2
RMF	0.4151	0.9791	0.8806
RMFJ	0.5230	0.9744	0.8837
RMFP	0.3184	0.9866	0.8840
RMFJP	0.3342	0.9859	0.8845

Finally, the distributions of the aggregate stand level attributes QMD and mean tree height for thinned stands were compared. The distributions for these stand attributes were compared by computing a p_{iae}-value for each pair of actual and simulated QMD and mean height distributions. Figure 3.22 and Figure 3.23 present the nonparametric probability density estimates for the actual and simulated QMD and mean height distributions for the testing scenario RMF. Visually, the actual and simulated distributions are quite similar for both QMD and mean height. The p_{iae}-values for the QMD and mean height distributions for scenario RMF are 0.9811 and 0.9777, respectively, indicating that the actual and simulated QMD and mean height distributions are almost identical, having approximately 98% of their probability mass in common. Table 3.12 presents the results for the other three testing scenarios as well as the results for scenario RMF for comparison. These results reinforce the results from the simple linear regressions and provide another indication of how well the tree list generation database methodology performs for thinned stands.

Table 3.12: Overall QMD and mean height distribution p_{iae}-values for actual and simulated thinned stands for all four scenarios. See Figure 3.22 and Figure 3.23.

Scenario	QMD p_{iae}-value	Mean height p_{iae}-value
RMF	0.9811	0.9777
RMFJ	0.9778	0.9755
RMFP	0.9794	0.9784
RMFJP	0.9812	0.9791

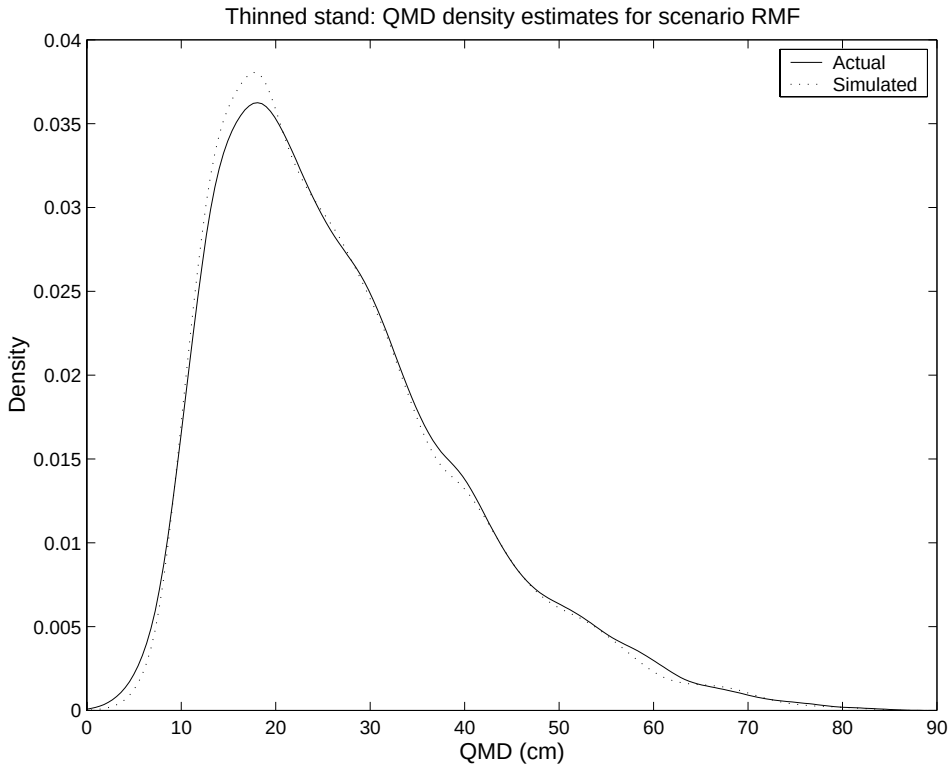

Figure 3.22: Nonparametric probability density function estimates for the actual and simulated QMD distributions for all thinned stands and scenario RMF. The p_{iae}-value for the actual *vs.* simulated QMD distributions is 0.9811. See Table 3.12 for the overall QMD distribution p_{iae}-values for the other scenarios.

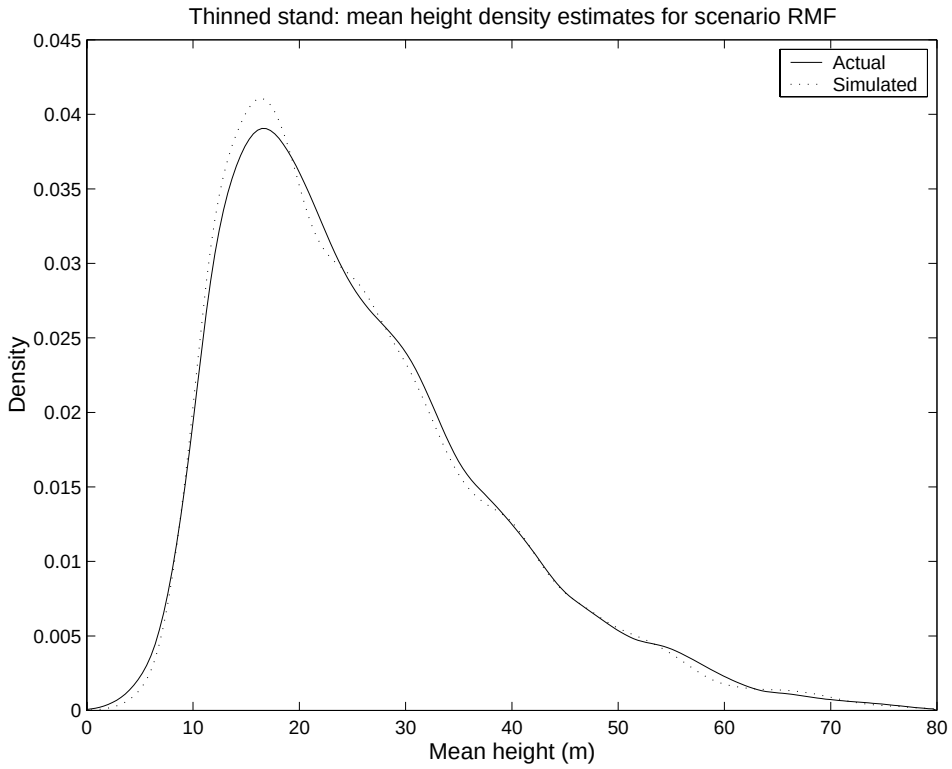

Figure 3.23: Nonparametric probability density function estimates for the actual and simulated mean height distributions for all thinned stands and scenario RMF. The p_{iae}-value for the actual *vs.* simulated mean height distributions is 0.9777. See Table 3.12 for the overall mean height distribution p_{iae}-values for the other scenarios.

3.3.5 Software testing procedures

When testing software there are two criteria that must be satisfied. First is that the software meets the requirements of its specification, i.e., does it simply work correctly. This is a verification the programs work correctly in a mechanical sense. Second is that the software actually solves the problem that it was intended to solve. This is a verification of the intent of the software. Both of these tasks are generally nontrivial to accomplish with any reasonably complex software system [23, 7, 19, 14]. When testing the tree list generation database software, both of these issues were addressed, but the latter was considered to be of much greater importance and therefore comprises the bulk of the testing. For more information on software testing and the software testing ideology adopted for the tree list generation database software, see the references cited.

The testing of the tree list generation database software consisted of four major phases. The first phase was concerned with the design and rapid prototyping of a tree list generation database to detect any early design flaws. The second phase was concerned with the correct operation of the subroutines and modules implemented for the tree list generation database subroutine library, commonly called unit testing. The third phase was concerned with integrating the modules in the tree list generation database subroutine library into the programs that access and manipulate a tree list generation database, commonly integration testing. The fourth phase consisted of validating that the tree list generation database programs worked, i.e., that the programs performed as expected in terms of their input and output, generating reasonable tree lists for the specified input stand descriptions. This last phase is the most important, as it demonstrates the effectiveness of the overall tree list generation database methodology, rather than a specific implementation, and encompasses the bulk of the testing.

In the first phase of testing a written specification of the capabilities desired for

the tree list generation database was created, and a basic partitioning of the tree list generation problem into more basic components and their implementation. These two activities are naturally coupled, so a prototype of the tree list generation database was implemented to determine design flaws. A more or less complete prototype of the tree list generation databse methodology for untreated stands was implemented using MATLAB [34] as a rapid prototyping environment. This rapid prototyping allowed modifications to the tree list generation database design to be performed easily, allowing a short turnaround time when problems with the initial tree list generation database specification were discovered. This phase may be thought of as generating a very detailed specification of the tree list generation problem with a particular software design as a solution. After the MATLAB prototype for untreated stands was completed and shown to be effective, the development of the tree list generation database Fortran 90/95 subroutine library was begun, and the second phase of testing was entered.

The second phase of testing the tree list generation database consisted of individual subroutine testing, to the extent possible or feasible, in the tree list generation database subroutine library. During this phase, the nominal input, output, and data processing activities for each subroutine were tested to verify correct behavior. In addition, groups of subroutines, or modules, that were required to operate together were also tested for their correct behavior. Once the individual subroutines and modules were demonstrated to be operating correctly, the third phase of testing began.

The third phase of testing the tree list generation database consisted of implementing the executable programs and integrating the subroutines and modules to perform the various tasks, the integration testing. Each program has had its nominal execution path thoroughly tested. The executable programs were also used to generate all of the tree lists used in this documentation and in phase four, the validation testing.

The final phase of the tree list generation database testing consisted of verifying

that the simulated tree lists generated by the system were reasonable, that is that the tree list generation methodology as implemented actually worked. This is the validation testing that comprises the bulk of this chapter.

3.4 Conclusions, caveats and other considerations

The tree list generation database methodology that has been described and implemented meets the objectives outlined in Section 3.1. Maximum use of the measured tree data is attained by storing the actual tree measurements. Trees are naturally represented and generated as multidimensional objects having a DBH, a height, and a species identifier. The nearest neighbor tree generation procedure provides the flexibility to represent a wide variety of stand types and tree size distributions. The trees generated for simulated stands are physically realizeable; they are generated from actual tree measurements, so they must be realizeable. The tree list generation database methodology represents different treatments in a consistent manner. Finally, the addition of new stand measurement data to a tree list generation database is simple and does not affect the overall stand classification.

The tree list generation database methodology that has been described and implemented appears to work quite well for both untreated and thinned stands. Simulated stands or tree lists generated using the tree list generation database methodology generally agreed with the actual stands they were generated to mimic at the individual tree level, defined by the DBH distribution, the height distribution, and the species composition. At least 80% of the simulated stands were found to be statistically indistinguishable from their actual, measured counterparts for both untreated and thinned stands using a critical p_{iae}-value of 0.75; the p_{iae}-value based misclassification or error rates were at most 20%. The regional, or overall, distributions of the stand level attributes QMD and mean height of the simulated and actual stands also agreed quite well, having approximately 98% agreement in their distributions for

both untreated and thinned stands, with the agreement in species composition also well in excess of 90%.

The results as presented are not broken down by stand type. The data used to populate the tree list generation database `tgdb1r00` consisted predominantly of pure Douglas-fir and pure western hemlock stands with much smaller percentages of Douglas-fir and western hemlock mixtures, see Table 3.2. Given the high degree of agreement between the actual and simulated stands, for both untreated stands and thinned stands, it seems unlikely that partitioning the data by stand type would give results that differ greatly from the aggregated results given here. A visual examination of the results broken down by stand type was performed to verify this claim. The visual assessment confirmed the statement that no benefit would be gained by partitioning the results by stand type. If the overall agreement were poorer than it is, splitting the results by stand type could have been useful for identifying situations where the tree list generation database methodology performed poorly.

In closing, there are two issues that remain to be considered. First, the total misclassification or error rates are conservative, i.e., too large. Given this, how much do they over estimate the actual total misclassification rates. A brief examination of this issue is presented in Section 3.4.1. Second is the use of the tree list generation database with forest growth and yield models and forest simulation systems. This will be addressed briefly in Section 3.4.2.

3.4.1 How conservative are the total error rates?

The 20% total misclassification rates observed for simulated untreated and thinned stands may be quite conservative, i.e., too large, for three reasons. First, the total misclassification rates were computed by performing a logical *or*ing of the three individual attribute misclassification events. This method of computing the total error rates must inflate their values when compared to computing an error simultaneously rate in the three dimensions DBH, height, and species simultaneously; the probability

mass in three dimensions is spread much more thinly than in one dimensional distributions, which decreases the probability that the actual and simulated stands will be statistically different. There is simply much more room to move around in two or more dimensions.

Second, in all of the p_{iae}-value based comparisons, the misclassification rates for tree height distributions were much greater than the misclassification rates for DBH distributions or species composition. An explanation of this phenomenon for both untreated stands and thinned stands is readily available: most of the tree heights are estimated from height-diameter models. The estimation of tree heights from a height-diameter model will dramatically reduce the variation of tree heights for a particular set of stand measurements relative the actual height variation for the stand at the time of its measurement. This supposition is partially supported by the fact that the height distribution misclassification rate is approximately 0.75% lower for the testing scenarios which jitter heights than for their non-jittered equivalents. For thinned stands there is an additional component that influences the height distribution misclassification rate: thinnings remove the undesirable, generally smaller, trees from a stand, which also reduces the variation in height, hence increasing the height distribution misclassification rate. This effect may be seen in the slightly higher height error rate for the thinned stands. If the height distribution misclassification rates were comparable to the DBH distribution error rates, the total error rates could be reduced by a factor of up to two, becoming approximately 10% to 15% misclassification.

Third, the critical p_{iae}-value may be too large, causing stands which are actually similar to be identified as different. The critical p_{iae}-value was determined through a sampling simulation based on the standard normal distribution which is strongly unimodal. In many forestry situations, both the DBH and height distributions may be multimodal or even somewhat uniform in shape. In both cases the critical p_{iae}-value for distinguishing between similar and dissimilar stands would be smaller than the value of 0.75 that was used.

The magnitude of the conservative effect on the misclassification rate caused by too large a critical p_{iae}-value may be examined using actual stand measurement data. Douglas-fir diameter data from the Stand Management Cooperative (SMC) were used to determine how small an appropriate critical p_{iae}-value may be for pure Douglas-fir stands. The diameter data consisted of individual tree DBH measurements from two to four permanent study plots on each of 30 SMC Type I pure Douglas-fir installations. The plots chosen from each installation had one to three commensurate measurement dates. The selected data provided 352 within stand comparison which were used to compute p_{iae}-values. The SMC Type I installations are all on commercial forest land, are naturally regenerated or planted juvenile stands, and have at least 90% Douglas-fir measured in stems per unit area. The site index at 50 years for the installations ranged from 25.9 m (85 ft) to 44.2 m (145 ft), with stand densities of approximately 250 tph (100 tpa) to 1850 tph (750 tpa), and breast height ages ranging from three years to 21 years.

The individual plot DBH distributions were compared for plots within the same installation and having the same measurement years to obtain p_{iae}-values. The p_{iae}-values were computed using the same procedures outlined in Section 3.3.1. A histogram of the computed p_{iae}-values is given in Figure 3.24. The histogram clearly indicates that the p_{iae}-value of 0.75 is conservative, and that a reasonable critical p_{iae}-value may be as small as 0.55, though there is little data for this small a p_{iae}-value. Thus, a critical p_{iae}-value of 0.70 or 0.65 may be more appropriate, and still somewhat conservative, choice for determining the similarity or difference of forest stands than the value of 0.75. Figure 3.25 plots the computed p_{iae}-values against the stand breast height ages. Notice the relatively uniform range of the p_{iae}-values between 0.70 and 1.00 over time. This would seem to indicate that the diameter distributions of the sample plots are at least maintaining their relative differences, as measured by the p_{iae}-value, over time. The Douglas-fir stands used for this comparison are quite young with relatively high site index values, and have not entered or

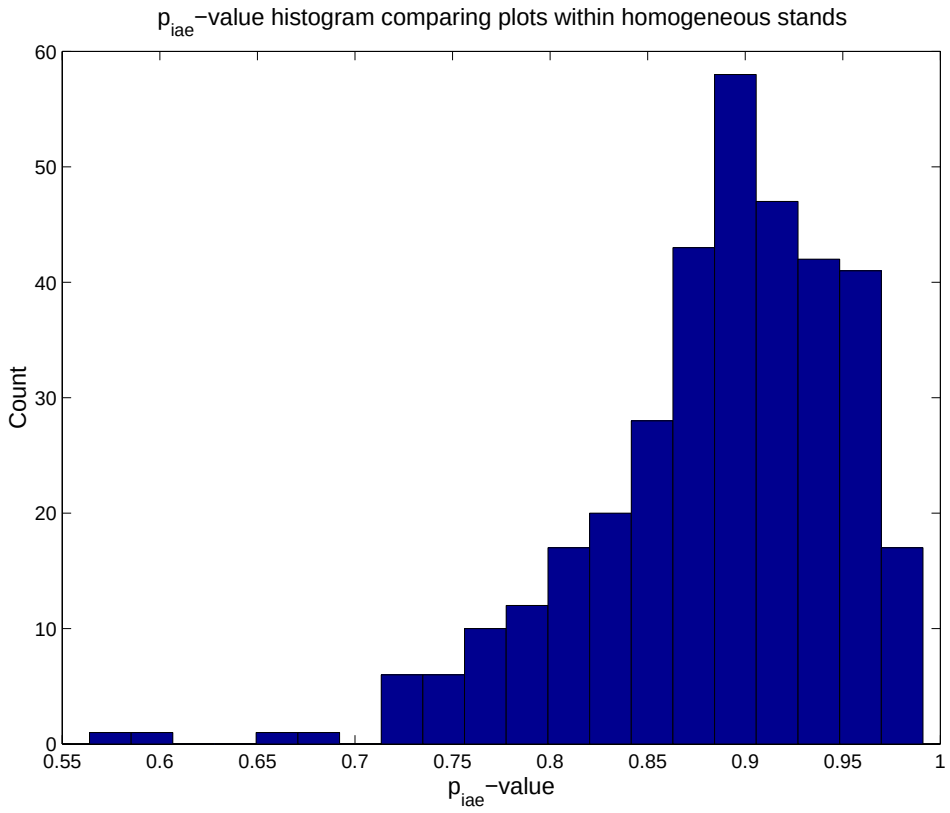

Figure 3.24: Histogram of DBH p_{iae}-values for actual stands. This plot indicates that the p_{iae} cutoff for comparing stands may be as low as 0.55, and that a value of 0.70 or even 0.65 may be more appropriate than the critical p_{iae}-value of 0.75. See Table 3.13 for the effect of lower p_{iae} cutoff values on the misclassification rates.

have just begun to enter a size differentiation stage of growth. As the size differention progresses within these stands, the critical p_{iae}-value for determining their similarity or difference would continue to decrease, possibly to a p_{iae}-value near 0.50.

The use of a lower critical p_{iae}-value would greatly reduce the observed total misclassification rates. See Table 3.13 for the misclassification rates that would be obtained for critical p_{iae}-values of 1.00 through 0.00 in steps of 0.05. Note in particular the very rapid decline of the misclassification rates to zero for critical p_{iae}-valuesbetween 0.75 and 0.50. A reduction of the critical p_{iae}-value from 0.75 to 0.70 cuts the total stand misclassification rate in half, reducing it to approximately 10%.

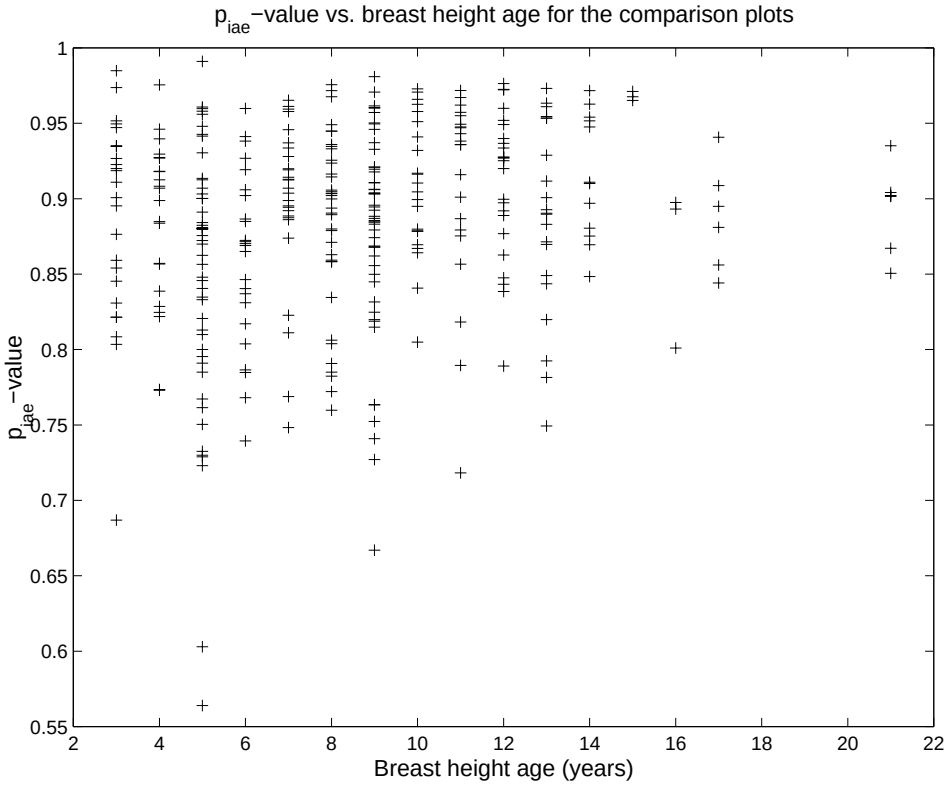

Figure 3.25: Plot of DBH p_{iae}-values *vs.* breast height age for actual stands. This plot indicates that the DBH p_{iae}-values may consistently maintain a fairly wide distribution through time.

A further reduction of the critical p_{iae}-value to 0.65, which seems justifiable given the DBH distribution comparison just performed, would again halve the total misclassification rate, reducing it to approximately 6%. If the critical p_{iae}-valueis as small is 0.50, which is a strong possibility, the total misclassification rate could be as small as 1%. Given this investigation, a critical p_{iae}-value of 0.65 seems reasonable, and obtians a total percentage of correctly classified simulated stands well in excess of 90%.

Table 3.13: Total misclassification rates for different critical p_{iae}-values and tree list generation scenarios. Multiplying the misclassification rate by 100 gives the percentage of simulated stands that would be considered different from the actual stands they were generated to mimic. Note the rapid decline in the misclassification rate as the critical p_{iae}-value is reduced.

	Untreated stands				Thinned stands			
Cutoff	RMF	RMFJ	RMFP	RMFJP	RMF	RMFJ	RMFP	RMFJP
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.9500	0.9791	0.9823	0.9781	0.9806	0.9743	0.9694	0.9705	0.9712
0.9000	0.7946	0.7955	0.7802	0.7846	0.7628	0.7601	0.7568	0.7601
0.8500	0.5285	0.5256	0.5145	0.5139	0.5032	0.4863	0.4917	0.4858
0.8000	0.3219	0.3258	0.3148	0.3166	0.3054	0.2982	0.2957	0.3000
0.7500	0.1972	0.1851	0.1929	0.1801	0.1800	0.1730	0.1804	0.1775
0.7000	0.1094	0.1071	0.1064	0.1052	0.0993	0.0977	0.1011	0.0980
0.6500	0.0603	0.0618	0.0584	0.0609	0.0523	0.0495	0.0550	0.0500
0.6000	0.0317	0.0313	0.0328	0.0305	0.0284	0.0243	0.0275	0.0259
0.5500	0.0157	0.0144	0.0157	0.0150	0.0140	0.0122	0.0144	0.0124
0.5000	0.0073	0.0056	0.0071	0.0056	0.0061	0.0045	0.0063	0.0063
0.4500	0.0029	0.0027	0.0035	0.0027	0.0029	0.0027	0.0036	0.0029
0.4000	0.0010	0.0010	0.0010	0.0010	0.0014	0.0018	0.0020	0.0016
0.3500	0.0006	0.0006	0.0008	0.0008	0.0011	0.0005	0.0007	0.0005
0.3000	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002
0.2500	0.0000	0.0000	0.0000	0.0000	0.0002	0.0000	0.0000	0.0000
0.2000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.1500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.1000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

3.4.2 Simulating forest stands using a tree list generation database

A typical forest growth and yield simulation would project a stand, generally represented by the trees on a single hectare (acre), some specified number of years into the future to obtain, for example, an estimate of the standing wood volume for harvest scheduling or other economic analyses. The results of this analysis would then be extended to the entire stand in order to make the appropriate business decisions. The tree list generation database has been designed to provide an easy way to generate forest stands as input to individual tree based forest growth and yield simulators, such as ORGANON [11], FVS [8, 35, 36] or TASS [20], or for use with other tree based forest stand or forest economics simulation systems.

In a classic forest growth and yield simulation setting, the simulation or growth projection of a single stand is used to represent all similar stands by scaling the simulation results to agree with the appropriate land area [5]. Unfortunately, this single simulation run does not provide any indication of the variation within stands that may be due to genetics, climate, differences in site quality, spatial distribution of the trees, or micro site variation. The tree list generation database may be used in conjunction with a forest growth and yield model or forest stand simulation system to obtain an indication of the possible variation in an actual stand through repeated simulations. Each simulation would use a different simulated stand, generated to match a particular set of stand attributes, as its input. The variation in the output from the repeated simulations should then approximate the variation for an actual stand after a similar time duration.

Figure 3.26 presents nonparametric probability density function estimates for the DBH and height distributions of an actual stand and ten simulated stands that were generated to mimic it using the tree list generation database `tgdb1r00`. The ten simulated stands for this comparison were generated with the default options of `TGRAND`, so the 100% species composition and jitter height options were both not used. Notice in

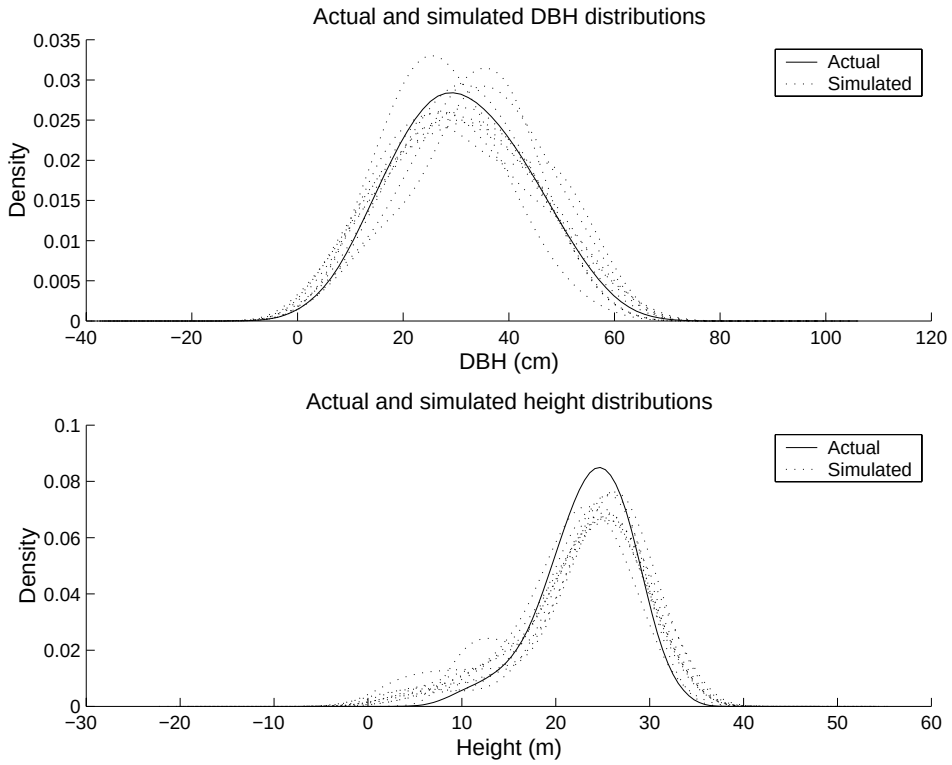

Figure 3.26: Plot of DBH and height density function estimates for 10 simulated stands for a particular stand and the DBH and height density function estimates for the actual stand. The p_{iae}-values for DBH, height, and species composition for each simulation are given in Table 3.14.

particular that the simulated stands seem to be generally reasonable approximations of the actual stand. This is supported by the DBH distribution, height distribution, and species composition p_{iae}-values obtained by comparing the actual stand with each of the simulated stands presented in Table 3.14. Nonzero values of the nonparametric density estimates in the figure for negative DBH or height values is an artifact of the probability density estimation technique; there were no negative tree diameters or heights generated.

There are several different interpretations that may be placed on the variation obtained by using different simulated initial stands to generate growth and yield predictions. The simulated initial stands may be thought of as accounting for local

Table 3.14: p_{iae}-values for DBH height, and species for 10 simulations of a particular stand. The DBH and height density function estimates for each simulation are given in Figure 3.26.

Simulation	DBH p_{iae}-value	Height p_{iae}-value	Species p_{iae}-value
1	0.9302	0.8745	0.9592
2	0.8985	0.9038	0.8980
3	0.9539	0.9293	0.9592
4	0.9531	0.8704	0.9592
5	0.8733	0.8619	0.9592
6	0.9406	0.9233	0.9592
7	0.9705	0.9441	0.8571
8	0.9142	0.8972	0.9184
9	0.8762	0.9100	0.9592
10	0.9236	0.9132	0.9592

climate variation, accounting to some degree for the influences of climate on tree and stand growth, since most growth and yield models do not include climate. The tree list generation database is particularly apropos for this interpretation since it makes use of actual tree measurements, which contain an integrated record of local climate, to generate simulated stands. Not every hectare (acre) in a stand is the same, due to genetics, tree spcing, and micro site variation. Simulated stands from the tree list generation database may be interpreted as representing the actual variability within stands, and hence giving, possibly, more reliable estimates of standing wood volume, or some other stand characteristic.

Given the simultaneous similarity and variation in the ten simulated stands, it seems likely that a particular, but small, set of stand attributes, e.g., QMD, average height, and species composition, could simultaneously be met at some point with repeated draws. Thus, if it was important to nearly exactly match a few particular numerical attributes of a stand, say to within $\pm 10\%$, this could be done by using the following algorithm.

1. Generate a simulated stand.

2. Compare the numerical values of the simulated attributes with those desired.

3. If the attributes agree to within the appropriate tolerances, stop; otherwise repeat the process.

If arbitrarily small tolerances are used it may not be possible to generate a stand with similar attributes, so some care must be taken when choosing a reasonable tolerance.

Finally, if a large gap exists in the data contained in a tree list generation database, a trusted forest growth and yield model or forest stand simulation system may be used to simulate the missing stand conditions and augment a tree list generation database. This would be acceptable if data were very expensive to collect, or if simulated stands for a very severe, and hence impractical to perform, set of silvicultural practices were desired. In either case, access to the resulting tree list generation database must be carefully controlled since it will contain a mixture of actual and simulated data, and results derived from such a database may not be reliable.

3.4.3 Limitations of the tree list generation database

As is true with any large or sufficiently complex data processing or simulation system, there are a variety of limitations affecting its use, and the tree list generation database is no different in this regard. The limitations of the tree list generation database may be broadly grouped into two categories: software limitations and data limitations, though there is some unavoidable overlap in these categories.

The primary software limitation of the tree list generation database is that it currently only supports pure Douglas-fir stands, pure western hemlock stands, and stands containing one of these two species as a dominant component in a mixed species stand. This limitation may be overcome by extending the tree list generation database software to include stands with arbitrary species composition. A secondary software limitation is that all of the tree data are stored, so a tree list generation database could become quite large. Given the availability low cost, high volume storage devices, this

limitation is of little practical concern. If the database size becomes large enough to degrade performance, more efficient sorting and searching algorithms may, however, need to be put into place.

The data limitations for the tree list generation database methodology should be obvious: if there are no data near a desired stand description, no simulated stand may be generated, or a generated simulated stand may not adequately match the attributes of the desired stand. For the near future this may be of some concern, but over time this limitation should effectively disappear as new stand measurement data are added to a tree list generation database to fill in gaps in its data coverage. This is particularly true for commercial forestry, where the range of possible stand conditions is rather tightly constrained. This limitation would be of much greater importance for a tree list generation database used to simulate natural forests, though this limitation may be overcome here as well by the collection of new data.

The tree list generation database methodology has been shown to be quite effective in simulating forest stands. Neither the software limitations of the current implementation nor the data limitations are insurmountable; they may both be overcome through the application of time and effort, and, should not impede the use of the tree list generation database methodology in practice.

Chapter 4

TREE LIST GENERATION DATABASE PROGRAMMER'S REFERENCE

A Fortran 95 library implementing the basic tree list generation database capabilities has been developed and tested. The library consists of 132 function points and approximately 39,000 lines of well documented Fortran 95. The library is approximately 99% portable to other Fortran 95 compilers, with all nonportable code isolated in subroutines or modules. The tree list generation database subroutine library was used to implement the eight tree list generation database programs.

This chapter presents a brief, one or two line, description of each subroutine, function, or entry point, broken down by library module. A brief description of each major subroutine group is also provided. These descriptions provide a basic reference manual for the tree list generation databse subroutine library. The executable programs are not documented here; the descriptions provided in Chapter 2 should provide adequate introduction.

4.1 Tree list generation database subroutine library

The tree list generation database subroutine library subroutines, functions, and entry points are briefly described in this section. The individual subroutine, function, and entry point descriptions are broken down by their function, or use, in the tree list generation database subroutine library. For low level details on the file formats or implementation, see the appropriate Fortran 95 source files; they are the sole repository of the specific details.

4.1.1 Subroutines for reading and writing measurement files

The subroutines listed define the interface for reading and writing tree list generation database stand measurement files. Stand measurement files are the primary input and output file format for actual and simulated stand measurement data.

`TGMFKMGR SUBROUTINE` This is the umbrella procedure to allow the sharing of data among the tree list generator measurement file keyword type conversion procedures. This procedure should not be called directly.

`TGMFKI2N ENTRY` in `TGMFKMGR` Tree list generator convert measurement file keyword integer ID code to its equivalent measurement file keyword name for a specified treatment.

`TGMFKN2I ENTRY` in `TGMFKMGR` Tree list generator convert measurement file keyword name to its equivalent integer ID code for a specified treatment.

`TGMFKNUM ENTRY` in `TGMFKMGR` Tree list generator get the number of measurement file keywords for a specified treatment.

`TGMFRD SUBROUTINE` Tree list generator, read a stand measurement file. Output units for numeric values are metric.

`TGMFWR SUBROUTINE` Write a tree list generator measurement file. Input units are assumed to be metric.

4.1.2 Subroutines for reading and writing description files

The subroutines listed define the interface for reading and writing tree list generation database stand description files. Stand description files are used to specify the stand attributes for generating a simulated stand or tree list.

`TGDFRD SUBROUTINE` Tree list generator, read a stand description file. Output units for numeric values are metric.

`TGDFWR SUBROUTINE` Write a tree list generator description file. Input units are assumed to be metric.

`TGDFWRT SUBROUTINE` Write a tree list generator description file template.

4.1.3 Subroutines for reading and writing schema files

The subroutines listed define the interface for reading and writing tree list generation database schema files. Schema files are used to specify the treatments, index parameters, and multidimensional histogram bin widths used in a tree list generation database.

`TGSCHRD SUBROUTINE` Tree list generator, read a schema file.

`TGSCHWR SUBROUTINE` Tree list generator, write a new schema file.

`TGSCHWRT SUBROUTINE` Tree list generator, write a new schema template file.

4.1.4 Binary search tree subroutines

The subroutines listed define the binary search tree interface used by the tree list generation database for classifying stands by their attributes. These subroutines maintain the index of unique multidimensional histogram bins defining the stand classification. This index is also used to determine the nearest neighbors for pooling tree data to create a canonical stand.

`TGBSTADD SUBROUTINE` Add the key and data portions of a double precision record to a binary search tree implemented on the DAS file architecture.

`TGBSTARD SUBROUTINE` Add a double precision record to a binary search tree file.

`TGBSTCLS SUBROUTINE` Close an open binary search tree file.

`TGBSTFRD INTEGER FUNCTION` Find a double precision record in a binary search tree file.

`TGBSTOPN SUBROUTINE` Open and initialize a new binary search tree file.

`TGBSTOPR SUBROUTINE` Open an existing binary search tree file for reading.

`TGBSTOPW SUBROUTINE` Open an existing binary search tree file for writing.

`TGBSTRDD SUBROUTINE` Read the data portion of a double precision record from a binary search tree implemented on the DAS file architecture.

`TGBSTRDK SUBROUTINE` Read the key portion of a double precision record from a binary search tree implemented on the DAS file architecture.

`TGBSTRRD SUBROUTINE` Read a double precision record from a binary search tree implemented on the DAS file architecture.

`TGBSTRSZ SUBROUTINE` Read the number of records and record sizes from a binary search tree implemented on the DAS file architecture.

`TGBSTWRD SUBROUTINE` Write a double precision record to a binary search tree implemented on the DAS file architecture.

`TGBSTWSZ SUBROUTINE` Write the number of records and record sizes to a binary search tree implemented on the DAS file architecture.

4.1.5 Bucket data mapping subroutines

The subroutines listed define the interface to the tree data bucket mapping used by the tree list generation database for mapping multidimensional histogram bins, or data buckets, to the individual tree data for that histogram bin. These subroutines provide an index from the multidimensional histogram bins to the individual tree data for each stand measurement file in the bins.

`TGBDMABD SUBROUTINE` Add bucket mapping data to the bucket mapping file.

`TGBDMCLS SUBROUTINE` Close an open bucket data mapping file.

`TGBDMFND SUBROUTINE` Search for a bucket index, returning the beginning and ending bucket data mapping records if the id is found in a bucket data mapping file.

`TGBDMIDX SUBROUTINE` Index the bucket data mapping records in a bucket data mapping file.

`TGBDMOPN SUBROUTINE` Open and initialize a new bucket data mapping file.

`TGBDMOPR SUBROUTINE` Open an existing bucket data mapping file for reading.

`TGBDMOPW SUBROUTINE` Open an existing bucket data mapping file for writing.

`TGBDMRDR SUBROUTINE` Tree list generator bucket data mapping read data record.

`TGBDMRMR SUBROUTINE` Read a bucket data mapping record from a bucket data mapping file.

`TGBDMSRT SUBROUTINE` Sort the bucket data mapping in a bucket data mapping file.

`TGBDMWDR SUBROUTINE` Tree list generator bucket data mapping write data record.

`TGBKTNN SUBROUTINE` Find the nearest buckets to the requested index parameter vector in a specified parameter index data set.

`TGBKTSCR DOUBLE PRECISION FUNCTION` Compute a bucket separation score for two sets of index parameter bucket centers and a a treatment description.

4.1.6 Tree measurement data subroutines

The subroutines listed define the interface to the tree measurement data file used by the tree list generation database for storing and retrieving the individual tree data for each stand. The random or simulated tree lists are also created using the `TGTMDRND` subroutine from this section.

`TGTMDAMD SUBROUTINE` Tree list generator tree measurement data add tree measurement data to a tree measurement data file.

`TGTMDCLS SUBROUTINE` Close an open tree measurement data file.

`TGTMDEXT SUBROUTINE` Extract tree measurement data for the specified bucket IDs from a tree measurement data file to a another tree measurement data file.

`TGTMDOPN SUBROUTINE` Open and initialize a new tree measurement data file.

`TGTMDOPR SUBROUTINE` Open an existing tree measurement data file for reading.

`TGTMDOPS SUBROUTINE` Open and initialize a new tree measurement data file as a scratch file.

`TGTMDOPW SUBROUTINE` Open an existing tree measurement data file for writing.

`TGTMDRMD SUBROUTINE` Tree list generator tree measurement data read tree measurement data from a tree measurement data file.

`TGTMDRMR SUBROUTINE` Tree list generator tree measurement data read a data record.

`TGTMDRND SUBROUTINE` Compute a set of random trees.

`TGTMDWMR SUBROUTINE` Tree list generator tree measurement data write a data record.

4.1.7 Database information subroutines

The subroutines listed define the interface to the tree list generation database information file. The information file contains the basic information about a tree list generation database, such as the treatments, index parameters used for each treatment, creation date, and other general information.

`TGINFCLS SUBROUTINE` Close an open tree list generator database information file.

`TGINFMGR SUBROUTINE` This is the umbrella procedure to allow the sharing of data among the tree list generator information file data structures and access procedures. This procedure should not be called directly.

`TGINFLDF ENTRY` in `TGINFMGR` Load a tree list generator information file into memory.

`TGINFUDF ENTRY` in `TGINFMGR` Update a tree list generator information file from memory.

`TGINFTRT ENTRY` in `TGINFMGR` Get or set the tree list generator information for a treatment.

`TGINFBKT ENTRY` in `TGINFMGR` Get or set the tree list generator primary or secondary bucket count for a treatment.

`TGINFSRC ENTRY` in `TGINFMGR` Get or set the tree list generator source file count for a treatment.

`TGINFTRS ENTRY` in `TGINFMGR` Get or set the tree list generator tree count for a treatment.

`TGINFLCK ENTRY` in `TGINFMGR` Get or set the tree list generation database lock status.

`TGINFTIM ENTRY` in `TGINFMGR` Get one of the tree list generator times: create time, modified time, or locked/unlocked time.

`TGINFHVT ENTRY` in `TGINFMGR` Return a logical flag indicating whether a treatment is available.

`TGINFOPN SUBROUTINE` Open and initialize a new tree list generation database information file.

`TGINFOPR SUBROUTINE` Open an existing tree list generation database information file for reading.

`TGINFOPW SUBROUTINE` Open an existing tree list generation database information file for writing.

4.1.8 Unit conversion subroutines

The subroutines listed define the interface to the tree list generation database units conversion procedures. The units conversion procedures provide a relatively painless interface for performing units conversions between the Imperial and metric systems of measurement.

`TGCHUNIT SUBROUTINE` Convert measurement units to another scale, either within a particular system of measurements, e.g., metric, or between systems of measurement, e.g., imperial and metric.

`TGMUNITS SUBROUTINE` This is the umbrella procedure to allow the sharing of data between the tree list generator measurement unit conversion procedures. This procedure should not be called directly.

`TGUNTI2N ENTRY` in `TGMUNITS` Tree list generator convert measurement units integer ID code to its equivalent measurement units name.

`TGUNTN2I ENTRY` in `TGMUNITS` Tree list generator convert measurement units name to its equivalent integer ID code.

`TGPRMCF SUBROUTINE` This is the umbrella procedure to allow the sharing of data among the tree list generator index parameter unit conversion procedures. This procedure should not be called directly.

`TGPCFI2M ENTRY` in `TGPRMCF` Tree list generator index parameter conversion factor for Imperial to metric conversion

`TGPCFM2I ENTRY` in `TGPRMCF` Tree list generator index parameter conversion factor for metric to conversion.

4.1.9 Index parameter keyword subroutines

The subroutines listed define the interface to the tree list generation database index parameter keywords and their associated integer ID codes. The index parameter names are used externally in files and reports, but integer ID codes are used internally for speed.

`TGPRMCTR SUBROUTINE` Compute the tree list generator bin centers for a parameter vector.

`TGPRMPK SUBROUTINE` Pack a full tree list generator vector of stand parameter values into the vector of parameter values that are in use.

`TGPRMTYP SUBROUTINE` This is the umbrella procedure to allow the sharing of data between the tree list generator parameter type conversion procedures. This procedure should not be called directly.

`TGPTI2N ENTRY` in `TGPRMTYP` Tree list generator convert parameter type integer ID code to its equivalent parameter type name.

`TGPTN2I ENTRY` in `TGPRMTYP` Tree list generator convert parameter type name to its equivalent parameter type integer ID code.

`TGPRMUPK SUBROUTINE` Unpack the stand parameter values that are in use into the a full tree list generator vector of vector of parameter values.

`TGPRMVEC SUBROUTINE` Compute an index parameter vector for a data set for a stand measurement.

4.1.10 Species mapping subroutines

The subroutines listed define the interface for using the tree list generation database species mapping files. Species mapping files are the means by which specific numeric ID codes may be associated with individual tree species.

`TGSPMMGR SUBROUTINE` This is an umbrella procedure that contains all of the species map manipulation procedures.

`TGSPMLD ENTRY` in `TGSPMMGR` Load a species map file into memory.

`TGSPCC2I ENTRY` in `TGSPMMGR` Convert a species two character code into its corresponding integer ID code.

`TGSPI2CC ENTRY` in `TGSPMMGR` Convert a species integer ID code into its corresponding two character code.

`TGSPI2CN ENTRY` in `TGSPMMGR` Convert a species integer ID code into its corresponding Common name.

`TGSPI2LC ENTRY` in `TGSPMMGR` Convert a species integer ID code into its corresponding 4-6 character code.

`TGSPI2LN ENTRY` in `TGSPMMGR` Convert a species integer ID code into its corresponding Latin name.

`TGSPLC2I ENTRY` in `TGSPMMGR` Convert a species 4-6 character code into its corresponding integer ID code.

`TGSPMNUM ENTRY` in `TGSPMMGR` Return the number of entries in the species map table.

`TGSPMRD SUBROUTINE` Read a tree list generator species map file.

`TGSPMWR SUBROUTINE` Write a tree list generator species map file.

4.1.11 Stand type subroutines

The subroutines listed define the interface to the tree list generation database stand types and their associated ID codes.

`TGSTTYP SUBROUTINE` This is the umbrella procedure to allow the sharing of data between the tree list generator stand type conversion procedures. This procedure should not be called directly.

`TGSTTI2N ENTRY` in `TGSTTYP` Tree list generator convert stand type integer ID code to its equivalent stand origin name.

`TGSTTN2I ENTRY` in `TGSTTYP` Tree list generator convert stand type name to its equivalent integer ID code.

4.1.12 Index parameter subroutines

The subroutines listed define the interface to the tree list generation database index parameters and their associated ID codes.

`TGIDXPRM SUBROUTINE` This is the umbrella procedure to allow the sharing of data AMONG the tree list generator index parameter type conversion procedures. This procedure should not be called directly.

`TGPRMI2N ENTRY` in `TGIDXPRM` Tree list generator convert index parameter integer ID code to its equivalent index parameter name for a specified treatment.

`TGPRMN2I ENTRY` in `TGIDXPRM` Tree list generator convert index parameter name to its equivalent index parameter integer ID code for a specified treatment.

`TGPRMNUM ENTRY` in `TGIDXPRM` Tree list generator get the number of index parameters for a specified treatment.

4.1.13 Stand origin subroutines

The subroutines listed define the interface to the tree list generation database stand origins and their associated ID codes.

`TGSTORIG SUBROUTINE` This is the umbrella procedure to allow the sharing of data between the tree list generator stand origin conversion procedures. This procedure should not be called directly.

`TGSTOI2N ENTRY` in `TGSTORIG` Tree list generator convert stand origin integer ID code to its equivalent stand origin name.

`TGSTON2I ENTRY` in `TGSTORIG` Tree list generator convert stand origin name to its equivalent integer ID code.

4.1.14 Treatment type subroutines

The subroutines listed define the interface to the tree list generation database treatment types and their associated ID codes.

`TGTRTTYP SUBROUTINE` This is the umbrella procedure to allow the sharing of data between the tree list generator treatment type conversion procedures. This procedure should not be called directly.

`TGTRTI2D ENTRY` in `TGTRTTYP` Tree list generator convert treatment type integer ID code to its treatment directory name.

`TGTRTI2N ENTRY` in `TGTRTTYP` Tree list generator convert treatment type integer ID code to its equivalent treatment type name.

`TGTRTN2I ENTRY` in `TGTRTTYP` Tree list generator convert parameter type name to its equivalent parameter type integer ID code.

`TGTRTHVN ENTRY` in `TGTRTTYP` Tree list generator return logical flag indicating whether a treatment is valid.

`TGTRTHVI ENTRY` in `TGTRTTYP` Tree list generator return logical flag indicating whether a treatment ID is valid.

4.1.15 Miscellaneous supporting subroutines

The subroutines listed are the miscellaneous supporting subroutines and functions used by the tree list generation database software. They include subroutines to compute standard forest measurements, e.g., QMD or basal area, and routines for performing various sorting and searching tasks. Some of the routines listed here also define some of the nonportable parts of the tree list generation database subroutine library.

BASARE SUBROUTINE Compute the individual tree basal area for a set of diameter at breast height measurements.

CHKFNM_1 SUBROUTINE Check a filename or path to be sure that it is valid.

DOMHT DOUBLE PRECISION FUNCTION Compute the dominant height for a list of trees. Dominant height is the average height of the 40 tallest trees per acre or the average height of the 100 tallest trees per hectare.

DSORTD SUBROUTINE Sort in place double precision records packed into a single dimensioned double precision array according to a dictionary ordering on the record components.

DSORTI SUBROUTINE Sort in place integer records packed into a single dimensioned integer array according to a dictionary ordering on the record components.

DSRTCD INTEGER FUNCTION Compare two arrays of double precision numbers using dictionary order on the array elements.

DSRTCI INTEGER FUNCTION Compare two arrays of integers using dictionary order on the array elements.

DUNI DOUBLE PRECISION FUNCTION This routine generates double precision uniform random numbers on [0,1).

DUSTAR ENTRY in DUNI This routine sets the seed for DUNI.

DUNIB ENTRY in DUNI This routine sets the mantissa precision for DUNI.

FUNIQI SUBROUTINE Find the unique integer values in an array of integers.

PRSCML_1 SUBROUTINE Parse a program command line into options, option values, if present, and command line arguments.

QMD DOUBLE PRECISION FUNCTION Compute the quadratic mean diameter for an array of diameter at breast height measurements.

TGBSLASH CHARACTER*(1) FUNCTION Return the backslash character, '
'.

TGCOMCHR CHARACTER*(1) FUNCTION Return the tree list generator comment character. The comment character appears as the first nonblank character on a line and indicates that the line is to be ignored.

TGGETMU SUBROUTINE Scan a Tree list generator input file looking for the keyword 'MEASUREMENT_UNITS' and returning its value.

TGGETTRT SUBROUTINE Scan a Tree list generator input file looking for the keyword 'TREATMENT_TYPE' and returning its value.

TGPTHCHR CHARACTER*(1) FUNCTION Return the tree list generator path element separator character. The path element separator is usually a backslash character, '
', or a forward slash character, '/', depending on the computing environment.

TGPTHFIX SUBROUTINE Fix the path element separation character in a path or filename. This subroutine allows either a forward slash, '/', or a backslash, '
', to be used as the path separation character.

TGTABCHR CHARACTER*(1) FUNCTION Return the tab character for use in the tree list generator. The tab character is not part of the FOrtran character set, so this function is provided to easily permit its use in programs.

`TOPHT DOUBLE PRECISION FUNCTION` Compute the top height for a list of trees. Top height is the average height of the 40 largest diameter trees per acre or the average height of the 100 largest diameter trees per hectare.

`TXTNLN INTEGER FUNCTION` This routine will scan the text file specified by FNAME, returning the number of lines in the file.

`TXTNLU INTEGER FUNCTION` This routine will scan the text file attached to the Fortran logical unit UNIT, returning the number of lines in the file.

`TXTOPS SUBROUTINE` Open a scratch text file for subsequent write access.

`VMEANG DOUBLE PRECISION FUNCTION` Compute the mean of a vector of double precision numbers.

`VSTDG DOUBLE PRECISION FUNCTION` Compute the standard deviation of a vector of double precision numbers.

`VSUMG DOUBLE PRECISION FUNCTION` Compute the sum of a vector of double precision numbers.

4.2 Future extensions and enhancements

This section contains a list of enhancements that could be made to the tree list generation database subroutine library. The list of possible enhancements is provided primarily as a guide to be used, or ignored, by the next programmer who modifies or enhances the tree list generation database subroutine library. The enhancements or modifications are not listed in any particular order.

- Add more information to the random stand measurement files when they are generated, such as a statistical summary of the trees used in the random tree

generation procedure and statistics for the bin characteristics of the selected data. Tree list generation databse version information should also be added to the files.

- Create a subroutine `TGTMDADD` for adding tree measurement data to a tree list generation database. This would provide a symmetry of interface for generating simulated stands and adding data. The program `TGADD` is currently the only way to add stand measurement data to a tree list generation database; there is no direct subroutine interface. For the near term, this should only prove a minor inconvenience – generating simulated stand via a single subroutine call was considered to be much more important.

- Hide the details of opening and closing the individual files used to represent a tree list generation database by creating a file manager. When working with a tree list generation database, it is the database that is either open or not open, not the individual files that comprise the database. The file manager would greatly simplify the high level interface.

- Add a delayed *database commit* for new data by staging the data before adding it to an existing tree list generation database. This entails creating a temporary tree list generation database that is always searched before the existing database is searched when adding data. In this case the existing tree list generation database would only be updated if all of the stand measurement files had been successfully added to the temporary database. The two database could then be merged to produce an updated database.

- For thinned stand measurement or description files, verify that the pre-thin stand density and basal area values are less than their post-thin values. This is not currently done.

- For each stand description keyword or index parameter keyword, define a set of valid input values. For categorical parameters with a small number of possible values, this could be an explicit list. For numeric values, this could be a valid range for the inputs. This information could be stored in a tree list generation database and used for input or output validation. This could possibly be done by creating a valid values entry point in the index parameter definition module.

- `TGDFRD`, `TGDFWR`, `TGMFRD`, `TGMFWR`, `TGSCHRD`, `TGSCHWR` all need do more stringent input and output value checking.

- Improve the error handling.

  - Not all errors are fatal errors.
  - Add a logging capability to the software so that errors during batch processing have a place to go, allowing processing to continue.

- Version information and modification information need to be added to the tree list generation database binary files. There is currently no version checking (there is only a single version). This would obviate some of the naming convention issues by potentially allowing a software detection of version or revision changes and changes in file contents.

- Add a way of annotating a tree list generation database. This capability exists in the NAIF SPICE DAS file architecture, but an interface to using it with a tree list generation database has not been implemented. This would allow a tree list generation database to carry a pedigree along with its data.

- Allow `TRUE` and `FALSE` as substitutes for `YES` and `NO`, respectively, for the appropriate index parameter keywords.

- Add default or required types for index parameters to prevent improper parameter type definitions. This would allow the detection of possible parameter type related problems, e.g., using a type of `INTERVAL` for a categorical variable.

- Allow the generation of dead but standing trees as well as live trees for a stand. Currently only live trees may be generated. This would involve a backwardly incompatible change by storing an additional data column with each tree indicating the status of the tree. It is well within the real of possibility.

- Along similar lines to the previous item, the definition of a tree as a multidimensional object needs to be generalized. Currently a tree may only be represented by a DBH, height, and species. This seems a bit too restrictive for the long run, however, and should be generalized. Missing value problems will need to be dealt with if this sort of generalization is performed.

- Permit arbitrary stand types to be defined by the dominant or codominant tree species. This would affect the `stand_type` and percentage by basal area and stems index parameters in a backwardly incompatible manner. This could be implemented by storing a stand type, `PURE`, `DOMINANT`, or `MIXTURE` with the species ID code for the tree species having the greatest percentage of the stand basal area. The interpretations of the `PURE`, `DOMINANT`, or `MIXTURE` would remain unchanged, but there would now be two index items representing stand type rather than a single stand type index parameter.

  - If this modification is made, the six percentage based index parameters for species composition should also be generalized. The same pattern will work: store the percentage and the tree species ID as a two part index item.

- Store the index parameter keywords used in the tree list generation database file

rather than coding them in a set of subroutines. This would increase the flexibility and overall applicability of the tree list generation database methodology, possibly, trivially, opening up applications in nonforestry fields.

  - This feature could even be extended to developing a mini- language for specifying index parameters based on input data types and simple commands for performing computations with data columns. The measurement file format appears to be quite flexible in this regard.

- The addition of a pre-thinning stand type could be useful. It is possible that during a thinning the stand type could change, e.g., from Douglas-fir dominant, $\geq$ 50% by basal area, to pure Douglas-fir, $\geq$ 80% by basal area. Most of the stands used did not have pre-thinning tree lists, so a pre-thin stand type could not be determined.

  - If this modification is made, the six percentage based index parameters for refining the species composition of a stand, should also be implemented, allowing the pre-thin stand to be as well defined as the post-thin stand.

- Addition of other index parameter keywords, such as GIS polygon ID, latitude and longitude, or other mechanisms to identify geographically similar data.

- Site index may prove to be problematic as a stand classification variable, so its continued use should in this regard considered carefully. Different species will have different reference ages, and for mixed species stands the definition of a site index value may be difficult or impossible. The use of an appropriate real stand measurement, such as top height, should be a more than adequate replacement for site index, given a multidimensional classification scheme such as that used in a tree list generation database.

- Improve the accuracy of the Imperial to metric and metric to Imperial conversions. The values used were obtained from convenient sources and are not necessarily the most accurate values available.

GLOSSARY

BACK-UP: A copy of a file or database to be used when catastrophe strikes the original. Always make one before adding data to a tree list generation database

COMMAND LINE ARGUMENT: A program argument specified on a command line to provide information, such as input or output file names, to a program. Command line arguments always follow the command line options.

COMMAND LINE OPTION: a program option identified by a short name preceded by a dash or minus sign that is specified on the command line for a program. Command line options typically modify the behavior of the program or provide additional information to the program. Command line options must precede any command line arguments.

CANONICAL STAND: This represents an aggregation of trees selected as the nearest neighbors to a desired set of stand attributes from a tree list generation database. The trees in a canonical stand are used as a lookup table to generate simulated tree lists that are similar to the specified stand attributes.

DATA BUCKET: Describes a multidimensional histogram bin defined by the index parameters and their interval widths as specified in a tree list generation database schema file.

DELIMITER: Something, often a single character, that indicates the beginning or ending of an argument or field in a file or on a command line. More generally, a delimiter is a field separator. Commonly used delimter characters are a space, comma, tab, or semi- colon.

DESCRIPTION FILE: A text file with a keyword equals value based format that provides a description of a stand of trees that is to be generated. The description of the stand is accomplished by defining appropriate values for the index parameter keywords for a particular tree list generation treatment and database.

IMPERIAL UNITS: System of meaurements based on the foot and pound.

INDEX PARAMETER: A stand attribute, e.g., stand type, site index, or trees per hectare or acre, that is used to create an index for the tree measurement data stored in the database. The index parameters are used to classify and select appropriate tree measurement data for the generation of a simulated stand of trees.

KEYWORD: An identifiable token in a file that may be used as a marker for an associated value. The keyword is usually a human understandable item.

KEYWORD EQUALS VALUE: A file format which makes use of an equal sign to associate a value, whether text or numeric, with a keyword.

LaTeX: A typsetting macro package developed by Leslie Lamport for computerized typesetting based on TeX [17]. The document prepartaion system used for this document.

MEASUREMENT FILE: A text file which contains the stand description keywords and tree measurement data for anobserved or simulated stand. A simulated, or random, stand measurement file may be distinguished from an actual stand measurement file by the word *random* which appears in its header comments.

METRIC UNITS: System of measurement based on the meter and kilogram.

PATH: A path is defined to be a string of path elements separated by a path separation character, commonly a forward slash '/' or a backslash '\', and ending in a valid path element or filename. A valid path may not end in a path separation character.

RANDOM STAND MEASUREMENT FILE: A text file which contains a set of stand description keywords and simulated tree measurement data for a simulated stand. A simulated, or random, stand measurement file may be distinguished from an actual stand measurement file by the word *random* which appears in its header comments.

SCHEMA FILE: A text file defining the index parameters and their attributes for a tree list generation database. A schema file is used to create a tree list generation database.

SPECIES MAP: A mapping from short species identification codes, e.g., `DF` or `WH`, to numeric codes for effecient storage and use of species information within the the tree list generation database and its supporting software.

SPECIES MAPPING FILE: A text file which defines a species mapping for a tree list generation database. The file contains five columns: a unique integer tree species ID code, a unique two (2) letter tree species code, a unique 4-6 letter tree species code, the common name for a species, and the Latin name for a species.

STAND DESCRIPTION KEYWORD: A keyword such as `SITE_INDEX`, `STAND_ORIGIN`, or `TREATMENT_TYPE`, which identifies information about a stand of trees that cannot be computed from a tree list. This information must appear in any stand measurement file that is to be added to a tree list generation database. The specific stand description keywords will vary depending on treatment type.

STAND DESCRIPTION FILE: See description file.

STAND INDEX PARAMETER: See index parameter.

STAND MEASUREMENT FILE: See measurement file.

TEX: A computerized typesetting system developed by Donald E. Knuth [16].

TREE LIST: A list of compatible diameter at breast height (DBH) and height measurements with an associated tree species.

VALUE: The text or numeric item on the right hand side of the equal sign in a keyword equals value format.

BIBLIOGRAPHY

[1] P.J. Bickel and K.A. Doksum. *Mathematical Statistics*. Holden-Day, 1978.

[2] G.S. Biging, T.A. Robards, E.C. Turnblom, and P.C. VanDeusen. The predictive models and procedures used in the forest stand generator (STAG). *Hilgardia*, 61(1):36 pp., 1994.

[3] Bruce E. Borders and William D. Patterson. Projecting stand tables: A comparison of the weibull diameter distribution method, a percentile-based projection method, and a basal area growth projection method. *Forest Science*, 36(2):413–424, 1990.

[4] David Bruce. Consistent height-growth and growth-rate estimates for remeasured plots. *Forest Science*, 27(4):711–725, 1981.

[5] Jerome L. Clutter, James C. Fortson, Leon V. Pienaar, Graham H. Brister, and Robert L. Bailey. *Timber Management: A Quantitative Approach*. Kreiger Publishing Company, reprint edition, 1992.

[6] William G. Cochran. The χ^2 test of goodness of fit. *Annals of Mathematical Statistics*, 23(3):315–345, 1952.

[7] John E. Dennis and Robert B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall, Inc., 1983.

[8] D.M. Donnelly. *Pacific Northwest coast variant of the forest vegetation simulator*. WO-Forest Management Service Center, USDA-Forest Service, Fort Collins, CO., 1997. Available on the Web.

[9] Kevin R. Gehringer. Nonparametric probability density estimation using normalized B-Splines. Master's thesis, The University of Tulsa, 1990.

[10] Kevin R. Gehringer and Richard A. Redner. Nonparametric probability density estimation using normalized *B*-splines. *Comm. Statist. Simulation Comput.*, 21(3):849–878, 1992.

[11] D.W. Hann, A.S. Hester, and C.L. Olsen. *ORGANON User's manual Edition 6.0*. Dept. Forest Resources, Oregon State University, Corvallis, OR 97331-5703, 1997.

[12] D. Kahaner, C. Moler, and S. Nash. *Numerical Methods and Software*. Prentice Hall, 1988.

[13] David Kahaner and George Marsaglia. double precision function duni. NIST Guide to Available Math Software, package NMS., 1988. Computes double precision uniform numbers on [0, 1).

[14] Brian W. Kernighan and Rob Pike. *The Practice of Programming*. Addison-Wesley Professional Computing Series. Addison-Wesley, 1999.

[15] James E. King. *Site Index Curves for Douglas-fir in the Pacific Northwest*. Number 8 in Weyerhaeuser Forestry Paper. Weyerhaeuser Company, 1966.

[16] Donald E. Knuth. *The TEX book*. Addison-Wesley, 1984.

[17] Leslie Lamport. *A Document Preparation System LATEX: User's Guide and Reference Manual*. Addison-Wesley, 2nd edition, 1994.

[18] George Marsaglia. Comments on the perfect uniform random number generator. Unpublished notes, Washington State University.

[19] Steve C. McConnell. *Code Complete : A Practical Handbook of Software Construction*. Microsoft Press, 1993.

[20] Kenneth J. Mitchell. *Dynamics and Simulated Yield of Douglas-fir*. Monograph 17. Forest Science, 1975.

[21] Gordon D. Nigh. The geometric mean regression line: A method for developing site index conversion equations for species in mixed stands. *Forest Science*, 41(1):84–98, 1995.

[22] Chadwick D. Oliver and Bruce C. Larson. *Forest Stand Dynamics*. John Wiley & Sons, Inc., update edition, 1996.

[23] P.J. Plauger. *The Standard C library*. Prentice-Hall, 1991.

[24] Richard A. Redner. Convergence rates for uniform B-spline density estimators. I. One dimension. *SIAM J. Sci. Comput.*, 20(6):1929–1953 (electronic), 1999.

[25] Richard A. Redner and Kevin Gehringer. Function estimation using partitions of unity. *Comm. Statist. Theory Methods*, 23(7):2059–2078, 1994.

[26] Richard A. Redner and Homer F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984.

[27] Marion R. Reynolds, Jr., Thomas E. Burk, and Won-Chin Huang. Goodness-of-fit tests and model selection procedures for diameter distribution models. *Forest Science*, 34(2):373–399, 1988.

[28] B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. London: Chapman Hall, 1986.

[29] Albert R. Stage and Melinda Moeur. Most similar neighbor: An improved sampling inference procedure for natural resource planning. *Forest Science*, 41(2):337–359, 1995.

[30] Lahey Computer Systems. *Lahey/Fujitsu Fortran 95 Express User's Guide*. Lahey Computer Systems, LF95 express, version 5.5 edition, 1999.

[31] Malcolm S. Taylor and James R. Thompson. A data based algorithm for the generation of random vectors. *Computational Statistics & Data Analysis*, 4:93–101, 1986.

[32] James R. Thompson and Richard A. Tapia. *Nonparametric Function Estimation, Modeling, and Simulation*. SIAM: Society for Industrial and Applied Mathematics, 1990.

[33] D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions*. John Wiley & Sons, 1985.

[34] The Math Works. *Using Matlab, Version 5.3*. The Math Works, Inc., 1999.

[35] W. R. Wykoff. *Supplement to the User's Guide for the stand PROGNOSIS model - Version 5.0*. USDA FS Intermountain For. and Range Exp. Stn., Ogden, UT. Gen. Tech. Rep. INT-208, 1986.

[36] W. R. Wykoff, N.L. Crookston, and A.R. Stage. *User's guide to the stand prognosis model*. USDA FS Intermountain For. and Range Exp. Stn., Ogden, UT. Gen. Tech. Rep. INT-133, 1982.

Appendix A

TGSUMRY OUTPUT FOR tgdb1r00

This appendix contains the contents of the summary file created by the TGSUMRY program command given in Figure 2.18 in Section 2.5.2. The TGSUMRY command executed was

```
PROMPT> tgsumry -file summary.txt tgdb1r00
```

and the contents of the file summary.txt appears below.

```
Tree list generation database summary.

Summary created on:  06/28/2000 at 13:06:44.32

Tree list generation database path: current directory
Tree list generation database name: tgdb1r00

Tree list generation database created on : 06/27/2000 at 17:17:27.82
Tree list generation database modified on: 06/28/2000 at 13:06:12.85

Tree list generation database lock status     : locked
Tree list generation database lock/unlock date: 06/28/2000 at 13:06:12.85

The units used in this summary are metric.

Number of treatments: 2

Treatment : UNTREATED

   Number of index parameters  : 7
```

```
   Number of tree data buckets      : 3139
   Number of source files           : 5209
   Number of tree records           : 573036

   Index parameters and parameter types:

   Parameter Name                   Used    Type      Width
   -------------------------------- ------- --------- ----------
   TREATMENT_TYPE                   YES        FIXED         N/A
   SITE_INDEX_50                    YES     INTERVAL        3.00
   STAND_TOTAL_AGE                  YES     INTERVAL        4.00
   STAND_ORIGIN                     YES        FIXED         N/A
   MEAN_DBH                         NO      INTERVAL        4.00
   MEAN_HEIGHT                      NO      INTERVAL        3.00
   QMD                              YES     INTERVAL        4.00
   TOP_HEIGHT                       NO      INTERVAL        3.00
   STAND_DENSITY                    YES     INTERVAL      200.00
   STAND_BASAL_AREA                 NO      INTERVAL        1.50
   PCT_DF_STEMS                     NO      INTERVAL       20.00
   PCT_WH_STEMS                     NO      INTERVAL       20.00
   PCT_OT_STEMS                     NO      INTERVAL       20.00
   PCT_DF_BASAL_AREA                NO      INTERVAL       20.00
   PCT_WH_BASAL_AREA                NO      INTERVAL       20.00
   PCT_OT_BASAL_AREA                NO      INTERVAL       20.00
   STAND_TYPE                       YES        FIXED         N/A
   NUMBER_OF_SPECIES                NO      INTERVAL        5.00

Treatment : THINNED

   Number of index parameters  : 12

   Number of tree data buckets      : 3716
   Number of source files           : 4440
   Number of tree records           : 237245

   Index parameters and parameter types:

   Parameter Name                   Used    Type      Width
   -------------------------------- ------- --------- ----------
   TREATMENT_TYPE                   YES     INTERVAL        2.00
   SITE_INDEX_50                    YES     INTERVAL        3.00
   STAND_TOTAL_AGE                  YES     INTERVAL        4.00
   STAND_ORIGIN                     YES        FIXED         N/A
   NUMBER_OF_THINNINGS              YES        FIXED         N/A
```

PRE_THIN_DENSITY	YES	INTERVAL	200.00
PRE_THIN_BASAL_AREA	YES	INTERVAL	1.50
PCT_STEMS_REMOVED	NO	INTERVAL	10.00
PCT_BASAL_AREA_REMOVED	YES	INTERVAL	10.00
YEARS_SINCE_TREATMENT	YES	INTERVAL	4.00
MEAN_DBH	NO	INTERVAL	4.00
MEAN_HEIGHT	NO	INTERVAL	3.00
QMD	YES	INTERVAL	4.00
TOP_HEIGHT	NO	INTERVAL	3.00
STAND_DENSITY	YES	INTERVAL	200.00
STAND_BASAL_AREA	NO	INTERVAL	1.50
PCT_DF_STEMS	NO	INTERVAL	20.00
PCT_WH_STEMS	NO	INTERVAL	20.00
PCT_OT_STEMS	NO	INTERVAL	20.00
PCT_DF_BASAL_AREA	NO	INTERVAL	20.00
PCT_WH_BASAL_AREA	NO	INTERVAL	20.00
PCT_OT_BASAL_AREA	NO	INTERVAL	20.00
STAND_TYPE	YES	FIXED	N/A
NUMBER_OF_SPECIES	NO	INTERVAL	4.00

Appendix B

TREE SPECIES COMMON AND LATIN NAMES

This appendix contains the tree list generation database species common and Latin names. These names were used with the tree list generation database `tgdb1r00`. This species mapping file is valid for the tree list generation database distributions `tlgv1r0d` and `tlgv1r0d`.

Table B.1: Tree list generation database tree species codes and names. The table presents a short two (2) character tree species codes, a long 4-6 character tree species codes, the common name for a tree species and the latin name for a tree species.

Short	Long	Common Name	Latin Name
BC	POTR	Black Cottonwood	*Populus trichocarpa*
BR	BEPA	Birch	*Betula papyrifera*
CA	RHPU	Cascara	*Rhamnus purshiana*
CH	PREM	Bitter Cherry	*Prunus emarginata*
DF	PSME	Douglas-fir	*Pseudotsuga menziesii*
DM	ACGL	Douglas Maple	*Acer glabrum*
DW	CONU	Western Flowering Dogwood	*Cornus nuttallii*
ES	PIEN	Englemann Spruce	*Picea engelmannii*
GC	CACH	Golden Chinkapin	*Castanopsis chrysophylla*
GF	ABGR	Grand Fir	*Abies grandis*
HD	HD	Other hardwoods	None
HW	CRDO	Hawthorn	*Crataegus douglasii*
HZ	COCO2	Hazelnut	*Corylus cornuta*
IC	LIDE	Incense Cedar	*Libocedrus decurrens*
KP	PIAT	Knobcone Pine	*Pinus attenuata*
LP	PICO	Lodgepole Pine	*Pinus contorta*

Table B.1: (continued)

MA	ACMA	Bigleaf Maple	*Acer macrophyllum*
MD	ARME	Madrone	*Arbutus menziesii*
MY	UMCA	Oregon Myrtle	*Umbellularia californica*
NF	ABPR	Noble Fir	*Abies procera*
OA	FRLA	Oregon Ash	*Fraxinus latifolia*
OO	QUGA	Garry Oak	*Quercus garryana*
OT	OT	Other species or mixtures	None
PC	CHLA	Port Orford Cedar	*Chamaecyparis lawsoniana*
PP	PIPO	Ponderosa Pine	*Pinus ponderosa*
RA	ALRU	Red Alder	*Alnus rubra*
RC	THPL	Western Red Cedar	*Thuja plicata*
RF	ABMA	California Red Fir	*Abies magnifica*
SA	ALCR2	Sitka Alder	*Alnus crispa*
SB	AMELA	Servicberry	*Amelachier spp.*
SF	ABAM	Pacific Silver Fir	*Abies amabalis*
SP	PILA	Sugar Pine	*Pinus lambertiana*
SS	PISI	Sitka Spruce	*Picea sitchensis*
SX	SALIX	Willow	*Salix spp.*
TO	LIDE2	Tan Oak	*Lithocarpus densiflorus*
VM	ACCI	Vine Maple	*Acer circinatum*
WF	ABCO	White Fir	*Abies concolor*
WH	TSHE	Western Hemlock	*Tsuga heteraphylla*
WL	LAOC	Western Larch	*Larix occidentalis*
WM	MYCA	Waxmyrtle	*Myrica californica*
WP	PIMO	Western White Pine	*Pinus monticola*
WY	TABR	Western Yew	*Taxus brevifolia*
YC	CHNO	Alaska Yellow Cedar	*Chamaecyparis nootkatensis*
AF	ABLA	Alpine fir	*Abies lasiocarpa*
BO	QUKE	Black Oak	*Quercus kelloggii*

Appendix C

A SAMPLE TREE LIST GENERATION DATABASE SPECIES MAPPING FILE

This appendix contains the tree list generation database species mapping file used to create the tree list generation database `tgdb1r00`. This species mapping file is valid for the tree list generation database distributions `tlgv1r0d` and `tlgv1r0s`.

```
%
% Tree list generation species mapping file.
%
% File Created: 08/05/1999 at 05:00:00
%
% DO NOT MODIFY THIS FILE UNLESS YOU KNOW WHAT YOU ARE DOING.
%
% This is the internal tree species ID to name mapping file for the
% tree list generation database software. This file should NOT be
% modified by any users of the program. Any additions or modifications
% made to this file may cause unpredictable results if the file is used
% with the tree list generation database software. This file is used to
% map two letter and four to six letter tree species ID codes into
% small integers for use in the tree list generation database software.
%
% It contains five columns:
%
%    1) The unique integer tree species ID code
%    2) The unique two (2) letter tree species code
%    3) The unique 4-6 letter tree species code
%    4) The common name for the species.
%    5) The Latin name for the species.
%
% The file is a comma delimited file, and the commas ARE NECESSARY.
%
```

```
% Comment lines and blank lines are ignored. Comment lines are lines
% whose first nonblank character is a percent sign (%).
%
% Note: If different integer ID codes are desired for a particular tree
%       list generation database, the integer codes in the file may be
%       changed and used in a NEW tree list generation database. Use of
%       the modified integer ID codes with an existing tree list
%       generation database will cause incorrect identification of the
%       species codes.
%

1,BC,POTR,Black Cottonwood,Populus trichocarpa
2,BR,BEPA,Birch,Betula papyrifera
3,CA,RHPU,Cascara,Rhamnus purshiana
4,CH,PREM,Bitter Cherry,Prunus emarginata
5,DF,PSME,Douglas-fir,Pseudotsuga menziesii
6,DM,ACGL,Douglas Maple,Acer glabrum
7,DW,CONU,Western Flowering Dogwood,Cornus nuttallii
8,ES,PIEN,Englemann Spruce,Picea engelmannii
9,GC,CACH,Golden Chinkapin,Castanopsis chrysophylla
10,GF,ABGR,Grand Fir,Abies grandis
11,HD,HD,Other hardwoods,None
12,HW,CRDO,Hawthorn,Crataegus douglasii
13,HZ,COCO2,Hazelnut,Corylus cornuta
14,IC,LIDE,Incense Cedar,Libocedrus decurrens
15,KP,PIAT,Knobcone Pine,Pinus attenuata
16,LP,PICO,Lodgepole Pine,Pinus contorta
17,MA,ACMA,Bigleaf Maple,Acer macrophyllum
18,MD,ARME,Madrone,Arbutus menziesii
19,MY,UMCA,Oregon Myrtle,Umbellularia californica
20,NF,ABPR,Noble Fir,Abies procera
21,OA,FRLA,Oregon Ash,Fraxinus latifolia
22,OO,QUGA,Garry Oak,Quercus garryana
23,OT,OT,Other species or mixtures,None
24,PC,CHLA,Port Orford Cedar,Chamaecyparis lawsoniana
25,PP,PIPO,Ponderosa Pine,Pinus ponderosa
26,RA,ALRU,Red Alder,Alnus rubra
27,RC,THPL,Western Red Cedar,Thuja plicata
28,RF,ABMA,California Red Fir,Abies magnifica
29,SA,ALCR2,Sitka Alder,Alnus crispa
30,SB,AMELA,Servicberry,Amelachier spp
31,SF,ABAM,Pacific Silver Fir,Abies amabalis
32,SP,PILA,Sugar Pine,Pinus lambertiana
33,SS,PISI,Sitka Spruce,Picea sitchensis
```

```
34,SX,SALIX,Willow,Salix spp.
35,TO,LIDE2,Tan Oak,Lithocarpus densiflorus
36,VM,ACCI,Vine Maple,Acer circinatum
37,WF,ABCO,White Fir,Abies concolor
38,WH,TSHE,Western Hemlock,Tsuga heteraphylla
39,WL,LAOC,Western Larch,Larix occidentalis
40,WM,MYCA,Waxmyrtle,Myrica californica
41,WP,PIMO,Western White Pine,Pinus monticola
42,WY,TABR,Western Yew,Taxus brevifolia
43,YC,CHNO,Alaska Yellow Cedar,Chamaecyparis nootkatensis
44,AF,ABLA,Alpine fir, Abies lasiocarpa
45,BO,QUKE,Black Oak,Quercus kelloggii
```

Appendix D

A SAMPLE TREE LIST GENERATION DATABASE SCHEMA FILE

This appendix contains the tree list generation database schema file used to create the tree list generation database `tgdb1r00`. This schema file is valid for the tree list generation database distributions `tlgv1r0d` and `tlgv1r0s`.

```
%
%  Tree list generator schema file.
%
%  Created on 08/08/1999 at 05:04:00
%
%  A properly formatted tree list generation database schema file is
%  used to create a new tree list generation database. A schema file
%  specifies the treatment types that are used in a tree list generation
%  database, and for each treatment type it specifies which index,
%  parameters are to be used and the attributes for each index parameter.
%
%  More details concerning the use of a tree list generatorschema file
%  and how to modify this schema template to create a valid tree list
%  generator schema file will be  added to the file at a future date.
%

MEASUREMENT_UNITS     = METRIC
TREATMENT_TYPE        = UNTREATED
   INDEX_PARAMETER    = SITE_INDEX_50
      USE_PARAMETER   = YES
      PARAMETER_TYPE  = INTERVAL
      INTERVAL_WIDTH  = 3.0
   END_INDEX_PARAMETER
   INDEX_PARAMETER    = STAND_TOTAL_AGE
      USE_PARAMETER   = YES
```

```
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 4.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = STAND_ORIGIN
   USE_PARAMETER   = YES
   PARAMETER_TYPE  = FIXED
END_INDEX_PARAMETER
INDEX_PARAMETER    = MEAN_DBH
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 4.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = MEAN_HEIGHT
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 3.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = QMD
   USE_PARAMETER   = YES
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 4.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = TOP_HEIGHT
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 3.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = STAND_DENSITY
   USE_PARAMETER   = YES
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 200
END_INDEX_PARAMETER
INDEX_PARAMETER    = STAND_BASAL_AREA
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 1.5
END_INDEX_PARAMETER
INDEX_PARAMETER    = PCT_DF_STEMS
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 20.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = PCT_WH_STEMS
   USE_PARAMETER   = NO
```

```
      PARAMETER_TYPE  = INTERVAL
      INTERVAL_WIDTH  = 20
   END_INDEX_PARAMETER
   INDEX_PARAMETER    = PCT_OT_STEMS
      USE_PARAMETER   = NO
      PARAMETER_TYPE  = INTERVAL
      INTERVAL_WIDTH  = 20.0
   END_INDEX_PARAMETER
   INDEX_PARAMETER    = PCT_DF_BASAL_AREA
      USE_PARAMETER   = NO
      PARAMETER_TYPE  = INTERVAL
      INTERVAL_WIDTH  = 20.0
   END_INDEX_PARAMETER
   INDEX_PARAMETER    = PCT_WH_BASAL_AREA
      USE_PARAMETER   = NO
      PARAMETER_TYPE  = INTERVAL
      INTERVAL_WIDTH  = 20.0
   END_INDEX_PARAMETER
   INDEX_PARAMETER    = PCT_OT_BASAL_AREA
      USE_PARAMETER   = NO
      PARAMETER_TYPE  = INTERVAL
      INTERVAL_WIDTH  = 20.0
   END_INDEX_PARAMETER
   INDEX_PARAMETER    = STAND_TYPE
      USE_PARAMETER   = YES
      PARAMETER_TYPE  = FIXED
   END_INDEX_PARAMETER
   INDEX_PARAMETER    = NUMBER_OF_SPECIES
      USE_PARAMETER   = NO
      PARAMETER_TYPE  = INTERVAL
      INTERVAL_WIDTH  = 5
   END_INDEX_PARAMETER
END_TREATMENT

TREATMENT_TYPE        = THINNED
   INDEX_PARAMETER    = SITE_INDEX_50
      USE_PARAMETER   = YES
      PARAMETER_TYPE  = INTERVAL
      INTERVAL_WIDTH  = 3.0
   END_INDEX_PARAMETER
   INDEX_PARAMETER    = STAND_TOTAL_AGE
      USE_PARAMETER   = YES
      PARAMETER_TYPE  = INTERVAL
      INTERVAL_WIDTH  = 4.0
```

```
END_INDEX_PARAMETER
INDEX_PARAMETER    = STAND_ORIGIN
   USE_PARAMETER   = YES
   PARAMETER_TYPE  = FIXED
END_INDEX_PARAMETER
INDEX_PARAMETER    = NUMBER_OF_THINNINGS
   USE_PARAMETER   = YES
   PARAMETER_TYPE  = FIXED
END_INDEX_PARAMETER
INDEX_PARAMETER    = PRE_THIN_DENSITY
   USE_PARAMETER   = YES
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 200.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = PRE_THIN_BASAL_AREA
   USE_PARAMETER   = YES
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 1.5
END_INDEX_PARAMETER
INDEX_PARAMETER    = PCT_STEMS_REMOVED
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 10.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = PCT_BASAL_AREA_REMOVED
   USE_PARAMETER   = YES
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 10.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = YEARS_SINCE_TREATMENT
   USE_PARAMETER   = YES
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 4
END_INDEX_PARAMETER
INDEX_PARAMETER    = MEAN_DBH
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 4.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = MEAN_HEIGHT
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 3.0
END_INDEX_PARAMETER
```

```
INDEX_PARAMETER    = QMD
   USE_PARAMETER   = YES
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 4
END_INDEX_PARAMETER
INDEX_PARAMETER    = TOP_HEIGHT
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 3.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = STAND_DENSITY
   USE_PARAMETER   = YES
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 200
END_INDEX_PARAMETER
INDEX_PARAMETER    = STAND_BASAL_AREA
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 1.5
END_INDEX_PARAMETER
INDEX_PARAMETER    = PCT_DF_STEMS
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 20.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = PCT_WH_STEMS
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 20.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = PCT_OT_STEMS
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 20.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = PCT_DF_BASAL_AREA
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 20.0
END_INDEX_PARAMETER
INDEX_PARAMETER    = PCT_WH_BASAL_AREA
   USE_PARAMETER   = NO
   PARAMETER_TYPE  = INTERVAL
   INTERVAL_WIDTH  = 20.0
```

```
   END_INDEX_PARAMETER
   INDEX_PARAMETER    = PCT_OT_BASAL_AREA
      USE_PARAMETER   = NO
      PARAMETER_TYPE  = INTERVAL
      INTERVAL_WIDTH  = 20.0
   END_INDEX_PARAMETER
   INDEX_PARAMETER    = STAND_TYPE
      USE_PARAMETER   = YES
      PARAMETER_TYPE  = FIXED
   END_INDEX_PARAMETER
   INDEX_PARAMETER    = NUMBER_OF_SPECIES
      USE_PARAMETER   = NO
      PARAMETER_TYPE  = INTERVAL
      INTERVAL_WIDTH  = 4.0
   END_INDEX_PARAMETER
END_TREATMENT
```

Appendix E

A TREE LIST GENERATION DATABASE SCHEMA TEMPLATE FILE

This appendix contains the schema template file for the tree list generation database distributions `tlgv1r0s` and `tlgv1r0d`.

```
%
%  Tree list generator schema file template.
%
%  File created: 03/24/2000 at 05:20:51.85
%
%  This file is a schema template file for the tree list generation
%  database software. As written, this schema template file is not a valid
%  input file for the tree list generation database software, but it may
%  be modified using any text editor and transformed into a valid tree
%  list generation database schema file. Read the rest of the file header
%  for information which describes how to transofrm this tree list
%  generation database schema template file into a valid tree list
%  generation database schema file.
%
%  A properly formatted tree list generation database schema file is used
%  to create a new tree list generation database. A tree list generation
%  database schema file specifies the treatment types that will be
%  contained in a tree list generation database, and for each treatment
%  type it specifies the index parameters to be used for each treatment
%  and the index parameter attributes that are to be used to classify each
%  stand measurement file that is stored in a tree list generation
%  databse. The index parameters that are defined in a tree list
%  generation database schema file, and used to create a tree list
%  generation database, are specify the attributes of a stand of trees
%  that is to be generated in the same manner as a query of a database us
%  used to select the data that is to be returned.
%
```

```
%  Each treatment that is defined by a tree list generation schema file
%  must contain the STAND_TYPE index parameter. The tree list generation
%  database software has been designed to work with forest stands that are
%  pure Douglas-fir, pure western hemlock, or Douglas-fir/western hemlock
%  mixtures. In order to properly classify the stand types in a tree list
%  generation database the STAND_TYPE index parameter must be present in
%  the schema file for each treatment that is defined.
%
%  Additionally, there are two sets of three index parameters for refining
%  the stand type by percent basal area by species and percent of stems by
%  species for Douglas-fir, western hemlock, and other tree species. If
%  one of the three index parameters from either of these sets is used
%  all three must be used to provide a consistent classification of the
%  stands that are stored in a tree list generation database. It is also
%  recommended that the same interval widths be used for each of the three
%  index parameters if either the percent basal area or percent stems by
%  species index parameters are used, again for consistency, though this
%  is not enforced.
%
%  More details concerning the use of a tree list generation database
%  schema file and how to modify a schema template file to create a valid
%  tree list generation database schema file may be found in the tree list
%  generation users guide.
%

MEASUREMENT_UNITS               = IMPERIAL | METRIC
TREATMENT_TYPE                  = UNTREATED
   INDEX_PARAMETER              = SITE_INDEX_50
      USE_PARAMETER             = YES | NO
      PARAMETER_TYPE            = FIXED | INTERVAL
      INTERVAL_WIDTH            = < number >
   END_INDEX_PARAMETER
   INDEX_PARAMETER              = STAND_TOTAL_AGE
      USE_PARAMETER             = YES | NO
      PARAMETER_TYPE            = FIXED | INTERVAL
      INTERVAL_WIDTH            = < number >
   END_INDEX_PARAMETER
   INDEX_PARAMETER              = STAND_ORIGIN
      USE_PARAMETER             = YES | NO
      PARAMETER_TYPE            = FIXED | INTERVAL
      INTERVAL_WIDTH            = < number >
   END_INDEX_PARAMETER
   INDEX_PARAMETER              = MEAN_DBH
      USE_PARAMETER             = YES | NO
```

```
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = MEAN_HEIGHT
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = QMD
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = TOP_HEIGHT
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = STAND_DENSITY
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = STAND_BASAL_AREA
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = PCT_DF_STEMS
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = PCT_WH_STEMS
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = PCT_OT_STEMS
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = PCT_DF_BASAL_AREA
```

```
      USE_PARAMETER            = YES | NO
      PARAMETER_TYPE           = FIXED | INTERVAL
      INTERVAL_WIDTH           = < number >
   END_INDEX_PARAMETER
   INDEX_PARAMETER             = PCT_WH_BASAL_AREA
      USE_PARAMETER            = YES | NO
      PARAMETER_TYPE           = FIXED | INTERVAL
      INTERVAL_WIDTH           = < number >
   END_INDEX_PARAMETER
   INDEX_PARAMETER             = PCT_OT_BASAL_AREA
      USE_PARAMETER            = YES | NO
      PARAMETER_TYPE           = FIXED | INTERVAL
      INTERVAL_WIDTH           = < number >
   END_INDEX_PARAMETER
   INDEX_PARAMETER             = STAND_TYPE
      USE_PARAMETER            = YES | NO
      PARAMETER_TYPE           = FIXED | INTERVAL
      INTERVAL_WIDTH           = < number >
   END_INDEX_PARAMETER
   INDEX_PARAMETER             = NUMBER_OF_SPECIES
      USE_PARAMETER            = YES | NO
      PARAMETER_TYPE           = FIXED | INTERVAL
      INTERVAL_WIDTH           = < number >
   END_INDEX_PARAMETER
END_TREATMENT

TREATMENT_TYPE                 = THINNED
   INDEX_PARAMETER             = SITE_INDEX_50
      USE_PARAMETER            = YES | NO
      PARAMETER_TYPE           = FIXED | INTERVAL
      INTERVAL_WIDTH           = < number >
   END_INDEX_PARAMETER
   INDEX_PARAMETER             = STAND_TOTAL_AGE
      USE_PARAMETER            = YES | NO
      PARAMETER_TYPE           = FIXED | INTERVAL
      INTERVAL_WIDTH           = < number >
   END_INDEX_PARAMETER
   INDEX_PARAMETER             = STAND_ORIGIN
      USE_PARAMETER            = YES | NO
      PARAMETER_TYPE           = FIXED | INTERVAL
      INTERVAL_WIDTH           = < number >
   END_INDEX_PARAMETER
   INDEX_PARAMETER             = NUMBER_OF_THINNINGS
      USE_PARAMETER            = YES | NO
```

```
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = PRE_THIN_DENSITY
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = PRE_THIN_BASAL_AREA
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = PCT_STEMS_REMOVED
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = PCT_BASAL_AREA_REMOVED
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = YEARS_SINCE_TREATMENT
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = MEAN_DBH
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = MEAN_HEIGHT
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = QMD
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = TOP_HEIGHT
```

```
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = STAND_DENSITY
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = STAND_BASAL_AREA
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = PCT_DF_STEMS
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = PCT_WH_STEMS
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = PCT_OT_STEMS
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = PCT_DF_BASAL_AREA
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = PCT_WH_BASAL_AREA
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
INDEX_PARAMETER             = PCT_OT_BASAL_AREA
   USE_PARAMETER            = YES | NO
   PARAMETER_TYPE           = FIXED | INTERVAL
   INTERVAL_WIDTH           = < number >
END_INDEX_PARAMETER
```

```
   INDEX_PARAMETER              = STAND_TYPE
      USE_PARAMETER             = YES | NO
      PARAMETER_TYPE            = FIXED | INTERVAL
      INTERVAL_WIDTH            = < number >
   END_INDEX_PARAMETER
   INDEX_PARAMETER              = NUMBER_OF_SPECIES
      USE_PARAMETER             = YES | NO
      PARAMETER_TYPE            = FIXED | INTERVAL
      INTERVAL_WIDTH            = < number >
   END_INDEX_PARAMETER
END_TREATMENT
```

Appendix F

THE SPICE SYSTEM - A BRIEF OVERVIEW

The Navigation Ancillary Information Facility (NAIF), located at the Jet Propulsion Laboratory in Pasadena, California, and acting under the directions of NASA's Office of Space Science, has built a data system, the SPICE system, to assist scientists in planning and interpreting scientific observations from space-borne instruments. The principal objective of this information system is to provide three dimensional geometric and other ancillary information used to plan space science missions and subsequently recover the full value of science instrument data collected and returned from these missions. This includes the correlation of individual instrument data sets with data from other instruments on the same or other spacecraft and the archiving of data in stable well documented file formats.

The primary SPICE data sets, often called *kernels* or *kernel files*. SPICE kernels are composed of navigation and other ancillary information that has been structured and formatted for easy access and correct use by the space science and engineering communities. SPICE kernel files are produced by the most knowledgeable sources of such information, usually located at a mission operations center. They must include or be accompanied by metadata – consistent with flight project data system standards – that provide pedigree and other descriptive information needed by prospective users.

SPICE kernel file types and their basic contents are summarized below. In the kernel type summaries an ephemeris is the three dimensional trajectory of a spacecraft, solar system body, or point in space, as seen from a particular point of view through time. The orbits of the planets around the Sun or the Moon around the Earth provide well known examples of ephemerides. The data types of the SPICE kernel files are

indicated by the letters of the SPICE acronym.

S- Spacecraft ephemeris, or more generally, location of an observer relative to another object, given as a function of time. These data are stored in a SPICE SPK, spacecraft and planet ephemeris kernel file.

P- Planet, satellite, comet, or asteroid ephemerides, or more generally, location of a target body, given as a function of time. These data are also stored in a SPICE SPK, spacecraft and planet ephemeris kernel file. The P-kernel also includes a second component containing certain physical, dynamical and cartographic constants for target bodies, such as size and shape specifications, and orientation of the spin axis and prime meridian. These target body physical and cartographic constants are found in the SPICE PCK file.

I- Instrument description kernel, containing descriptive and operational data peculiar to a particular scientific instrument, such as mounting alignment, internal timing relative to the spacecraft clock, and field-of-view model parameters. This instrument information is contained in the SPICE IK file.

C- Pointing kernel, containing a transformation traditionally called the C-matrix that provides time-tagged pointing (orientation) angles for a spacecraft structure upon which scientific instruments are mounted. Attitude data are contained in the SPICE CK file.

E- Events kernel, summarizing mission activities-both planned and unanticipated. Events data are contained in the SPICE EK file set which consists of three components: Science Plans, Sequences, and Notes.

The ephemeris data for spacecraft and target bodies are normally combined in a single file called the SPICE SPK file, though this is not necessary. An SPK file could

contain only target ephemerides, or only a spacecraft ephemeris, depending on the context and the particular data access requirements.

In addition to the standard SPICE kernel files, there are several miscellaneous kernel files, the spacecraft clock kernel (SCLK) file and the leapseconds kernel (LSK) file. These files are used to facilitate time conversions between various time measurement systems.

Also available is the SPICE Database Kernel (DBK) file. This provides a simple, portable relational database functionality used for several purposes, such as the Sequence component of the E-kernel, star catalogs, image catalogs and SPICE kernel file management. The Spice Database Kernel may also be used by customers in place of a commercial product when wide portability and low cost are key requirements.

Perhaps the SPICE acronym should have been SPICES, with the final **S** standing for Software. The SPICE system includes the NAIF Toolkit, a large collection of allied software. The principal component of this Toolkit is a library of portable subroutines for reading and writing the kernel files, calculating the most common observation geometry parameters, and performing a wide variety of text and data processing tasks. Users integrate these SPICE Toolkit subroutines into their own application programs to compute observation geometry parameters and related information where, and as, needed. The NAIF Toolkit was originally implemented in ANSI FORTRAN 77, now a subset of Fortran 90/95, but is becoming available in the C language as well. A C++ version is planned for some time in the future.

NAIF has designed the kernel file and software Toolkit architectures with portability and multimission application as principal goals. In addition, because extensive software documentation and examples are provided with the Toolkit, with a reasonable learning effort the software can be confidently used by the full spectrum of the NASA-supported space science community.

A flight project's mission operations center concentrates on producing, cataloging and distributing complete and accurate kernels on a timely basis. Kernel updates

should be made promptly if/as improved data sources become available.

Users may order those SPICE kernels of interest fo use in application programs hosted at their home sites to compute needed geometric and related ancillary information. Users may even update some kernels and produce their own versions of other kernels to support their own analyses or to provide their colleagues with any improvements in ancillary information resulting from their work.

Each flight project generally delivers well documented copies of all SPICE kernels and a set of developed Toolkit software to the appropriate permanent archive facility, assuring ready availability of this data for future use. Ideally this archive is open to the international community of scientists and engineers. User-produced kernels may also be similarly archived. For NASA planetary missions, the NAIF node of NASA's Planetary Data System (PDS) is the archive site.

Because generic ephemerides for most solar system target bodies are available at the NAIF PDS node, SPICE is frequently used for mission design and for planning solar system and stellar observations. In these cases the observer could be a *predict* spacecraft ephemeris produced by a mission design organization, a terrestrial telescope or a user-provided instrument location. In some cases predict versions of other SPICE kernels are also made to help simulate a full data processing system. With this flexibility scientists may use SPICE throughout the experiment life cycle – from mission planning to detailed observation design to instrument data analysis and finally to correlation of results other instruments on a spacecraft or other missions.

A large set of core SPICE components is in place. Extension and adaptation of this core system to encompass broader functionality and to meet specific needs of new projects is an ongoing endeavor. This work will include provision of some broadly useful application programs and development of additional kernel types. Examples of new applications are: adaptation of SPICE for landers and rovers, and, use of SPICE to point and tune NASA's Deep Space Network antennas to track all customer spacecraft.

The SPICE system is, or soon will be, used on numerous national and international space missions, such as Galileo, Mars Global Surveyor, Near Earth Asteroid Rendezvous, Mars Surveyor 98, Stardust, and Cassini. Limited data from older missions such as Voyager, Viking, and Pioneer have been restored and converted into SPICE data formats. While the principle use of SPICE has been in the planetary science discipline, astrophysics, space physics and even Earth science projects are also using this technology.

In addition to providing SPICE technology to NASA and various international space missions the NAIF Group serves as the Ancillary Data Node of NASA's Planetary Data System. In this role NAIF provides permanent archive, data distribution, and consultation functions for planetary project ancillary data sets.

July, 1998