Growth Modelling for Mixed Tropical Forests

Denis Alder

ODA Forestry Research Programme Project R4676
Final Camera-Ready Proof
Oxford, 16 May 1995

Tropical Forestry Paper 30
ISBN 0 85074 135 1
ISSN 0141-9668

ACKNOWLEDGEMENTS

This manual has been produced as part of Research Project R4676 of the Forestry Research Programme, funded by the United Kingdom Overseas Development Administration. The author gratefully acknowledges this assistance, and especially the work of Anne Bradley, Administrator to the FRP, and Mr. H.L. Wright, Programme Manager.

During the preparation of this manual, the author has been privileged to participate in a number of ODA-supported projects which have contributed the opportunity to test ideas and have provided sample data sets used in the various examples.

The GHAFOSIM model, and the various examples of analyses drawn from Ghana old-series permanent sample plots, where developed as a result of the author's participation as a consultant in the ODA/Ghana Forest Inventory and Management project between 1988 and 1992. Special thanks are due to the Chief Conservator of Forests, Mr. John Francois, for the facilities offered and permission to use this material.

The CAFOGROM model has been developed as part of the ODA/Brazil Silviculture Research Project, located at the *Centro do Pesquisa Agropecuaria da Amazonia Oriental* (CPATU), Belém, Brazil. The author's collaborators on that assignment included Dr. J. Natalino Macedo Silva, Lia Cunha de Oliveira, and R. Parente de Oliveira, who have been responsible over the years for establishing, collecting and analysing data from the CPATU plots at Tapajos. Some of this data is presented in the examples in this text and is implicit in the growth functions and starting conditions of the CAFOGROM model. The author would like to thank Dr. JNM Silva for permission to use this material.

Dr. M.D. Swaine of the University of Aberdeen, and the Botany Department, University of Ghana, Legon, made available the data sets for the Kade plots used in a number of the examples. The use of this material is gratefully acknowledged.

The draft of the manual was reviewed by Jerry Vanclay, Svend Korsgaard, Nick Brown and Douglas Shiel. Many thanks are due to them for their time and constructive comments and suggestions.

SUMMARY

This manual is directed at the analysis of permanent sample plot (PSP) data from mixed tropical forests to produce growth and yield models. It assumes some knowledge of matrix algebra, statistical methods, and computer programming. Its emphasis is on the practical technique of data analysis and model building. It is divided into five sections. The first covers definition of modelling terms and selection of strategy. It recommends a deterministic empirical strategy as most appropriate for forest planning applications from conventional PSPs and inventory data, and describes two basic approaches: Diameter class projection and cohort modelling.

The second section covers the basic analysis of PSP data to the point of calculating increments and competition indices. Using XBASE examples, methods are described for data entry, error checking, listing data, drawing plot maps, converting incompatible measurements to a common basis, merging multi-year data sets, and transformation of year-per-record and multi-year record data formats. The compilation of stand-level statistics, increment, and competition indices in a record format suitable for regression analysis is described. Detailed competition index calculation methods are given for spatial and non-spatial measures, including competition influence overlap and overtopping basal area. Reduction of large data sets by cross-tabulation is exemplified.

The third section covers diameter class projection in both its classical and matrix algebra formulations. The homologies between Usher, Markov and other matrix model formulations are explored. The use and relevance of the de Liocourt ratio is discussed. Spreadsheet examples (using Lotus 1-2-3) are given of classical stand projection and matrix models. A BASIC program for stand projection is compared with the spreadsheet examples. The transformation of data structures from raw PSP data to transition matrix are shown. The weaknesses of diameter class projection methods are discussed, including insensitivity to stand density, and the assumption of uniform increments within a class. The GHAFOSIM program is described as a case study including automatic updating of transition matrices and initial stand tables from PSP data and inventory plots, and the refinement of the method to include different crown classes and stand density interactions. Diameter class methods are considered suitable for aggregated projections as initial approximations in the early stages of forest management or for a broad sectoral overview of potential yield.

The fourth section covers cohort modelling methods and the derivation of increment, mortality, and recruitment functions. Cohort modelling is considered to be the optimal strategy for forest planning applications. An outline structure of a cohort model is discussed and exemplified using the author's CAFOGROM model for Brazil. The analysis of diameter increment is described. Log transformation of increment data is demonstrated graphically and considered appropriate. Basal area increment versus diameter increment are explored, and several possible robust equations indicated. The methods of including competition index are shown, and the problem of the weak residual covariance with competition once diameter has been accounted for discussed. Data sets

from forests of limited treatment range may not permit the evolution of adequate competition models. Site effects are treated as categorical variables, and methods of analysis exemplified using general linear models.

Mortality calculation and adjustment for interval measurement period are demonstrated using compounding of the survival rate. An interactive XBASE dialog to calculate mortality rates from PSP data is given. Mortality is a binomial proportion, and treatment of confidence intervals is described. The logistic transformation is appropriate for analysis. Methods of comparing species differences in mortality rates are shown.

Recruitment and regeneration are defined. Usually only recruitment data is available; regeneration modelling is more complex and requires specialized data. Calculation from raw PSP data is demonstrated. Prediction of recruitment is recommended as a two-part process. The quantity of recruits can be related to a level of stand disturbance or basal area loss in a previous time period, with the lag depending on defined recruitment diameter. The distribution of recruits by species is best not handled by regression due to problems of additivity and covariance, but can be simply managed by creating look-up tables of species proportions for different disturbance classes.

The simulation of harvesting, logging damage, and silvicultural treatment is discussed, with empirical methods and examples for estimating logging damage functions in relation to logging intensity. A case study of specific increment, mortality, recruitment, and logging damage functions is presented using the CAFOGROM model.

The final part of the manual deals with model validation. Procedures for residual analysis of individual functions are discussed, with examples of common pathologies (heteroscadacity, bias, lack-of-fit). Residual analysis applied to complete models is discussed. Methods for testing models by analysing their behaviour in relation to expected forest dynamics and yields are considered and exemplified using CAFOGROM. The application of a model as part of a forest management information system is described briefly, including linkages to data from temporary inventory plots, a species database, and forest stand locations and previous treatment history.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

LIST OF EXAMPLES

1 INTRODUCTION

1.1 Objectives and scope

This manual is designed to provide methods for analysing data from permanent sample plots (PSPs) for tropical mixed forests (TMF). This analysis is directed towards the construction of forest growth models. The central objective of these is to improve the sustained yield management of TMF. This requires models that can relate regulatory factors such as minimum felling diameters, the numbers of seed trees to be retained, and the intervals between fellings, to the extractable yields from the forests.

It is intended to be a practical guide to the research worker who may not hitherto have had very much experience in this field. Small computer programs, spreadsheets, and statistical analyses are elaborated and discussed in detail. Very often, in the author's experience, there is a gap between theoretical comprehension of an analytical concept and the ability to apply it to provide a useful result working with real data. The present work aims to fill that gap. *It does not attempt to provide a standard set of computer programmes for analysing permanent sample plot data.*

This concentration on detailed technique necessarily limits the scope of the modelling methods that can be covered. The focus is on two empirical approaches: Diameter class projection methods (including matrix models), and cohort/growth function models. The variety of conceptual methods potentially available for constructing growth models is now quite broad. A perspective and detailed review is given in Vanclay (1994) in a book that is recommended as a companion volume to the present work.

1.2 Prerequisites: skills, resources, and data

The user of this book will probably be engaged in research, either in a forest research institute, on a development project, or at a university as a staff member or graduate student. The central objective of that research will be the development of a growth and yield model for a particular region of tropical mixed forest.

In terms of human skills, certain prerequisite knowledge is assumed, including:

- An elementary knowledge of matrix algebra, calculus, and statistical methods, especially of multiple regression techniques.

- Computer skills including familiarity with a major spreadsheet package such as Lotus 1-2-3, Microsoft Excel, or Borland Quattro. Of particular importance is a basic knowledge of programming with a language from the XBASE group, such as Borland DBASE, Microsoft FoxPro, or Nantucket Clipper. For the material on cohort modelling, a more advanced knowledge of a block-structured programming language (especially C) will be found helpful, but is not assumed for most of the examples.

- A detailed familiarity with forest mensuration and the analysis of data from forest and tree measurements.

The material resources required to undertake the work described in this manual include especially a good computer of modern desktop standard, and able to run appropriate computer software. The most important software packages required will be:

- *The database package.* The use of an XBASE language has been assumed, and Microsoft FoxPro is recommended. SQL systems may be available or may be preferred, but will not allow most of the examples in this book to be run directly. The SQL SELECT statement is however available in some implementations of XBASE (eg. FoxPro 2.6) and has been used in some examples.

- *A spreadsheet package.* The author has used Lotus 1-2-3, which includes matrix algebra functions. In principle, other packages such as Excel, Quattro, or SuperCalc are also suitable. No particular recommendation is made, as all modern spreadsheets provide sufficient functionality for the purposes of this text.

- *A statistical/graphics package.* The author has used SYSTAT, which is comprehensive and especially effective for graphical analysis. Other major packages such as SAS, SPSS, Genstat or GLIM may be partially or wholly suitable.

- *A programming language such as Basic, Pascal, or C.* Several examples are given in Basic, whilst Borland Turbo C 2.0 is used as the programming language for the CAFOGROM example discussed in sections 4 and 5. Construction of a complete dynamic simulation model will require such a language, although the final choice is a matter of user preference.

The availability of suitable data is also essential. It is common for academic theses and research papers on growth modelling to be published on the basis of the analysis of very small data sets, often comprising one or two plots. Such work is of almost no use for forest management, as the coverage of the data is too limited. This manual is directed towards the analysis of large sets of PSP data, probably comprising a hundred or more plots, all of which have been measured at least twice over a reasonably long interval (3-5 years as a minimum).

Standards and techniques for the establishment of PSPs in tropical mixed forest are given in Vanclay (1994:79-102) or, in more detail, in Alder & Synnott (1992).

1.3 **Outline and usage**

This manual comprises five parts, including this introduction. Their coverage, and suggested usage, are shown in Table 1. Section 2 covers the essential details of entering PSP data into a suitable database format, with criteria for database selection and design. It also details how derived variables including tree

Name	Basis of model	Suitability	Examples
Yield model	Yield functions fitted empirically to time series data on standing forest height, basal area, and volume.	Classical approach for uniform forests. In TMF, requires plots with known logging history over a range of periods since last felling. Heterogeneity and dynamics of uneven-aged forest make this type of model inaccurate, but useful for crude estimates of allowable cut if approrpiate data available.	Canonizado (1979)
Diameter class projection	Forest characterised by stand table, with trees 'moved' between size classes at growth rates determined from PSP data.	Classical method for uneven-aged forests. Useful for highly aggregated models operated via computer programs or spreadsheet models. Complicated and less efficient than cohort approach for more detailed modelling.	Kofod (1982), Alder (1990), Korsgaard (1992)
Matrix model	Forest stand table treated as mathematical vector. Growth, mortality, and recruitment formulated as a matrix of coefficients used to multiply stand vector to achieve new values.	Mathematically elegant, but in practice too rigid a formulation. Linear form of matrix model does not allow feedback between stand density and growth.	Usher (1966), Buongiorno & Michie (1980)
Cohort model	Forest described by a list of cohorts, each comprising species, mean tree size (diameter or basal area), and tree condition. Functions are applied to cohorts to calculate growth and mortality. New cohorts are added to emulate recruitment.	Efficient and highly flexible, this approach is most suitable for construction of detailed models for forest management. Method may be too complex where data is crude or a quick, highly aggregated, simple projection method is sought.	Vanclay (1989a), Alder (1994)
Gap model	Forest described by a list of 'gaps', or spatial units typically about 0.1 ha. Each gap is modelled via a list of trees, with competition being determined relative to the largest. Seedling development usually via a cohort submodel until numbers small enough for trees to be represented individually.	Represents spatial heterogeneity and local processes of competition well. An intrinsically single-tree approach that therefore uses Monte Carlo functions and hence gives random outcomes. Complex to implement, does not easily map onto conventional inventory plots in TMF, and produces random outcomes, hence more suitable as ecological research tool than management model.	Botkin et al. (1972), Urban et al. (1991)
Tree position model	Individual trees represented by position, so that competition effects calculated from spatial relationships, relative heights and crown development of adjacent trees.	Spatial competition indices are very demanding of computer time, hence such models are usually limited to a single stand of at most a few hectares. Not suitable for forest management, as cannot give extensive spatial coverage. Useful as research tool to explore inter-tree competition and silviculture of stands.	Ek & Monserud (1974), Young & Muetzelfeldt (1993)
Pipe model	A mathematical model of tree growth based on assumption that each unit of new leaf area is supported by constant sectional area of active xylem (sapwood).	A process-driven approach to tree modelling that requires approriate destructive sampling to determine key empirical coefficients. Has potential for the future, but does not use conventional PSP data.	Waring et al. (1982)
Carbon balance model	A mathematical model of forest growth that considers leaf area, light extinction rates, and respiration rates to determine balance between production and utilization of cellular carbon.	A process-driven approach, suitable for global and regional estimation of gross wood production without respect to species or mensurational characteristics. Requires exotic parameters and specialized sampling, not conventional PSP data.	Valentine (1989)

Table 2 *Types of forest growth model, and their suitability for management and other applications*

increment and tree competition indices can be calculated. Individual tree data for large numbers of PSPs may be too voluminous to process directly, so methods are described for condensing data via cross-tabulation without losing essential information for regression analysis.

Section 3 covers the traditional methods of stand projection, and their more modern formulation as transition matrix models based on either the Usher matrix or Markov matrix paradigms. A case study of the structure of a matrix model developed for sector planning applications in Ghana, GHAFOSIM, is given to show a complex example of a stand projection model in the real context of TMF management. Examples using spreadsheets are also given for simple diameter class projection and transition matrix models.

Section 4 describes the concepts and structure of cohort/growth function models, as probably the most effective strategy for developing models for forest management applications. It describes in some detail regression and linear modelling methods for fitting diameter increment, recruitment and mortality models. It also describes methods for modelling logging and silvicultural treatments. It uses the CAFOGROM cohort model (Alder, 1994) as an example for program structure, and presents as a case study the functions used in that model.

Section 5 considers the validation of growth functions and growth models by residual analysis and the behavioural study of complete systems. It discusses how a model may be applied in the context of a forest management information system, and the issue of compatibility for inventory data.

In Appendices A and B, the reader is provided with an order form to obtain a demonstration diskette for the CAFOGROM model, and a complete listing of the C program code for that model. This is referred to in section 4 in discussing the construction of cohort models.

An index is provided at the end of the book which lists specialised terms, symbolic names and acronyms used, and author citations. This will be found to be helpful as the first entry in the index will normally give the page reference on which the item in question is defined.

The essential intention of the manual is that the various procedures given should be worked through by the user with their own data sets. Two basic modelling strategies are considered in depth: Firstly, methods based on diameter class projection, including matrix models; and secondly methods that use algebraic functions and a cohort representation of the forest stand. Section 2, which considers basic data storage and preparation, is common to both approaches. Other methods of modelling forest stands, such as tree position models and process-oriented concepts, are not considered, except that some calculation methods for tree-position competition indices are given.

1.4 Selection of a growth model design strategy

Forest growth models can be considered to fall into a number of groups, each of which is characterised by a central design paradigm, and each of which is most suitable for particular situations. These are listed in Table 2. The selection of modelling paradigm will be driven mainly by currently or potentially available data and the objectives for which the model is to be constructed. The present manual confines itself to those models which can be developed from conventional PSPs, and which are to be applied to forest management planning for timber production and yield regulation. This includes diameter class projection models, matrix models, and cohort models. Matrix models are regarded as a specialised case of diameter class projection, with the two approaches being treated together in Section 3. The design and development of cohort models is described in some depth in Section 4.

There are some generic terms used in relation to forest modelling designs that are used in Table 2 and elsewhere in the text that require definition. These are as follows:

Aggregation/Dis-aggregation: An aggregated model is one which lumps together heterogeneous information about the forest and assumes that an accurate projection can be made by projecting the mean of the data. For example, TMF is typically vary variable over any given distance scale. It is an assumption that a stand projection based on the stand table for a forest of say 100,000 ha will in any way correspond to the mean of individual projections of stand tables on compartments of say 250 ha each on the same forest. Similarly, species may be aggregated into groups and projections made on the basis of those groups. These may or may not correspond to the mean projections of individual species.

Generally, yield, diameter class projection and matrix models are used as aggregated models, with species and spatial descriptive data being highly grouped. Cohort models are suitable for application to relatively small forest units, corresponding to a compartment or felling block. Gap models are intrinsically disaggregated, and are very suitable for representing spatial heterogeneity of operations. The other classes of model can generally be only applied to single stands due to the limitations of the specialized parameters and data processing time required.

Deterministic, stochastic, or Monte Carlo: A deterministic model is one which uses functions to predict mean outcomes, and takes no account of the probability distribution of random processes. A stochastic model incorporates descriptive functions of probabilities. A Monte Carlo model uses stochastic functions in conjunction with a random number generator to produce random outcomes, so that each model run will in principle give different results. Stochastic models may yield deterministic or Monte Carlo (random) outcomes, depending on how they are treated within the computer program.

Name	Basis of model	Suitability	Examples
Yield model	Yield functions fitted empirically to time series data on standing forest height, basal area, and volume.	Classical approach for uniform forests. In TMF, requires plots with known logging history over a range of periods since last felling. Heterogeneity and dynamics of uneven-aged forest make this type of model inaccurate, but useful for crude estimates of allowable cut if approrpiate data available.	Canonizado (1979)
Diameter class projection	Forest characterised by stand table, with trees 'moved' between size classes at growth rates determined from PSP data.	Classical method for uneven-aged forests. Useful for highly aggregated models operated via computer programs or spreadsheet models. Complicated and less efficient than cohort approach for more detailed modelling.	Kofod (1982), Alder (1990), Korsgaard (1992)
Matrix model	Forest stand table treated as mathematical vector. Growth, mortality, and recruitment formulated as a matrix of coefficients used to multiply stand vector to achieve new values.	Mathematically elegant, but in practice too rigid a formulation. Linear form of matrix model does not allow feedback between stand density and growth.	Usher (1966), Buongiorno & Michie (1980)
Cohort model	Forest described by a list of cohorts, each comprising species, mean tree size (diameter or basal area), and tree condition. Functions are applied to cohorts to calculate growth and mortality. New cohorts are added to emulate recruitment.	Efficient and highly flexible, this approach is most suitable for construction of detailed models for forest management. Method may be too complex where data is crude or a quick, highly aggregated, simple projection method is sought.	Vanclay (1989), Alder (1994)
Gap model	Forest described by a list of 'gaps', or spatial units typically about 0.1 ha. Each gap is modelled via a list of trees, with competition being determined relative to the largest. Seedling development usually via a cohort submodel until numbers small enough for trees to be represented individually.	Represents spatial heterogeneity and local processes of competition well. An intrinsically single-tree approach that therefore uses Monte Carlo functions and hence gives random outcomes. Complex to implement, does not easily map onto conventional inventory plots in TMF, and produces random outcomes, hence more suitable as ecological research tool than management model.	Botkin et al. (1972), Urban et al. (1991)
Tree position model	Individual trees represented by position, so that competition effects calculated from spatial relationships, relative heights and crown development of adjacent trees.	Spatial competition indices are very demanding of computer time, hence such models are usually limited to a single stand of at most a few hectares. Not suitable for forest management, as cannot give extensive spatial coverage. Useful as research tool to explore inter-tree competition and silviculture of stands.	Ek & Monserud (1974), Young & Muetzelfeldt (1993)
Pipe model	A mathematical model of tree growth based on assumption that each unit of new leaf area is supported by constant sectional area of active xylem (sapwood).	A process-driven approach to tree modelling that requires approriate destructive sampling to determine key empirical coefficients. Has potential for the future, but does not use conventional PSP data.	Waring et al. (1982)
Carbon balance model	A mathematical model of forest growth that considers leaf area, light extinction rates, and respiration rates to determine balance between production and utilization of cellular carbon.	A process-driven approach, suitable for global and regional estimation of gross wood production without respect to species or mensurational characteristics. Requires exotic parameters and specialized sampling, not conventional PSP data.	Valentine (1989)

? which a or b ? a both.

Table 2 *Types of forest growth model, and their suitability for management and other applications*

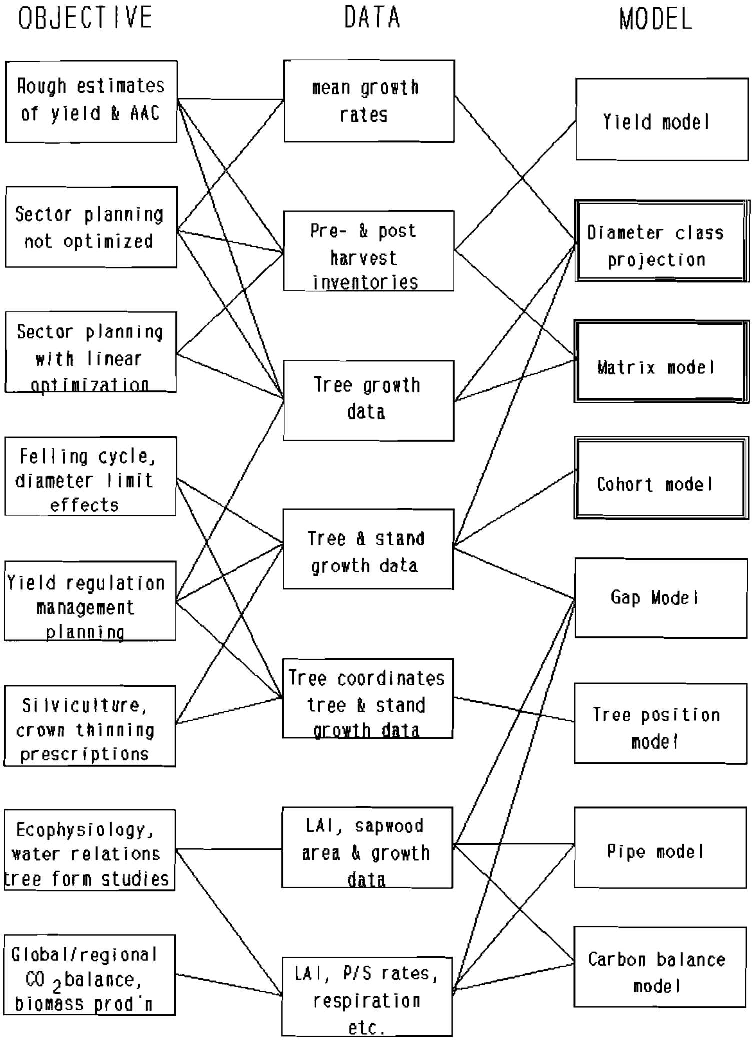

OBJECTIVE
DATA
MODEL
Rough estimates of yield & AAC
Sector planning not optimized
Sector planning with linear optimization
Felling cycle, diameter limit effects
Yield regulation management planning
Silviculture, crown thinning prescriptions
Ecophysiology, water relations tree form studies
Global/regional CO 2 balance, biomass prod'n
mean growth rates
Pre- & post harvest inventories
Tree growth data
Tree & stand growth data
Tree coordinates tree & stand growth data
LAI, sapwood area & growth data
LAI, P/S rates, respiration etc.
Yield model
Diameter class projection
Matrix model
Cohort model
Gap Model
Tree position model
Pipe model
Carbon balance model

Figure 1 ***Linkages between objectives, available data, and modelling strategy***

Generally, Monte Carlo models are not suitable for direct use in forest management planning, as random outcomes are somewhat confusing when, for example, trying to list a schedule of compartments to be felled by years. Models which represent individual trees, such as gap and tree position models, necessarily require Monte Carlo functions for processes that mediate the existence or non-existence of trees. Such processes include natural mortality, harvesting, and recruitment. Consequently such models, are *ipso facto* not very suitable for forest management planning.

Cohort, matrix and diameter class projection models can incorporate random processes as probability distributions whilst yielding deterministic outcomes. This is because they are not applied to individual trees, and probabilities can be treated directly as proportions.

Empirical or Process-derived: An empirical model is one whose coefficients have no direct correspondence to any physical, chemical, or physiological constants. The coefficients are determined by some process of function minimization such as least-squares regression or maximum liklihood analysis. The form of the functions is more or less arbitrary, developed from a graphical review of the data.

A process-derived model is one that is built mathematically from considerations of the physical, chemical, and biological processes involved. The coefficients of such models will ideally be resolvable to constants that are determinable from basic physical, chemical or physiological measurements.

Clearly, process-derived models represent an ideal to be striven towards. However, forests in general, and tropical mixed forests especially, are very complex ecosystems. The goal of a purely process-driven model of forest growth remains very much in the future. The methods treated in this manual are strictly empirical and designed to use conventional forest measurements.

Figure 1 shows the interaction or linkages between modelling objectives, available data, and modelling strategy in another way. Those models which are explicitly considered in this manual (diameter class, matrix, and cohort models) are shown as bold outline boxes. These can be built from conventional PSP data, or, in the case of diameter class projection, from previously analysed mean growth and mortality rate summaries. The more sophisticated and process-oriented modelling strategies require non-conventional sampling and measurement methods, and are definitely to be associated more with research than forest management objectives.

2 DATA ENTRY, EDITING AND PRELIMINARY PROCESSING

2.1 Introduction

Figure 2 shows some of the topics covered in this chapter. The principle theme is the management and preliminary processing of PSP data, using a standard database package.

Database design is the first stage in setting up data management procedures. In the ideal world, this would precede the design of field forms and procedures for making plot measurements. In practice it is more often necesssary to process a pre-existing body of data, either entering it for the first time on computer from paper archives, or transferring it onto modern systems from files on magnetic tapes or similar media.

Data entry and editing are very straightforward when a database package is used. A good conformance between field sheets and the computer display screen assists this process. Data entry requires also some thought as to how data will be checked and errors detected. A number of examples of suitable approaches are considered in detail.

The successive measurements made on PSPs over the years will normally be stored in separate computer files and merged prior to analysis. It will also be necessary to list out printed copies of the raw PSP data for checking and documentation. Plot maps are very useful, especially for locating trees during remeasurement and can be produced directly from a database package using plotter commands. Examples of DBASE commands and programs for these various operations are presented.

Because of the long periods over which measurements are made on PSPs, incompatibilities often arise between successive measurements due to changes in standards, such as the switch from imperial to metric units, or the change in the height of measurement, the instruments used (calipers versus tapes), or the tree numbering or species coding system. Methods are discussed for standardizing data. In tropical forestry, the problem of a changing point of measurement due to buttress formation is particularly relevant.

Once incompatibilities between successive data sets have been resolved, then increment can be calculated. The procedures for this, and for the generation of files for export to statistical packages for regression analysis are discussed. It will often be that the dataset is too massive to be analysed tree by tree, and the method of reducing the data to more manageable proportions using cross-tabulation are described.

Finally, methods are discussed for calculating the tree competition indices that are central to the analysis of increment and mortality functions. There are a great variety of such indices and they are not reviewed here in detail. Instead a number of basic methods are described in terms of actual XBASE programs to perform the calculations and produce output files ready for regression analysis.

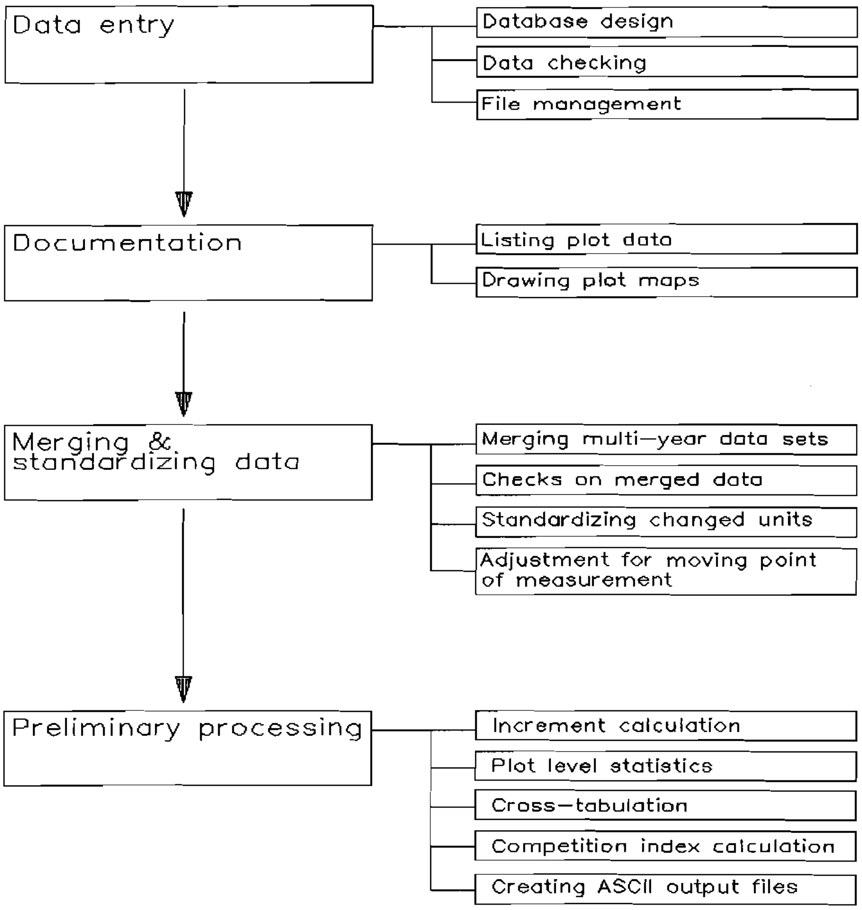


Figure 2 *The logical sequence of operations in the preliminary processing of PSP data*

2.2 Database selection and design

2.2.1 *Selection of a database package*

It is recommended that a standard database package is used for data management and reduction. Compared with the alternative of entering data in ASCII form as text files, there are several advantages:

- Time is saved by avoiding the need to write specialised programs to perform standard operations such as screen formatting, editing, checking, listing, file indexing, merging, sorting, and so on. These processes are all implicit in standard database packages, usually to a high level of optimization and functionality.

- Discipline and formalism is imposed on the design of file structures which saves later confusion and eases access to and use of data.

- It is much easier to document the database and related manipulations and pass on knowledge of the system to other users or programmers.

There are many comprehensive database systems available. The selection of the most appropriate will in the end be determined by existing standards and user preferences in a particular institution. Generally the following points should be given consideration:

- The package should support a command language that can be compiled to executable programs to run independently of the main package.

- The command language should support large multi-dimensional arrays. This is necessary for many data manipulations.

- Conformity to either the XBASE or SQL language sets will simplify training, documentation, compatibility, and upgrades to new systems.

The XBASE language group is that derived from the original dBASE III program, and includes packages such as Borland dBASE IV, Microsoft FoxPro, Nantucket Clipper, Wordtech Systems DBXL and others. SQL (Structured Query Language) is derived from IBM mainframe packages and is now extensively supported and defined by an ANSI standard. There are numerous implementations. R:BASE is a good example. SQL systems offer a more powerful and elegant language than XBASE, but actual implementations may be less well-refined than their XBASE equivalents on MS-DOS platforms and are comparatively very expensive.

Both FoxPro and DBASE IV allow the use of some SQL statements, especially the SELECT statement.

Standards other than XBASE and SQL exist such as Borland Paradox that are widely popular and may be perfectly suitable choices, especially if there is a prior investment in skills and software.

Modern programming languages such as Microsoft's Visual Basic or Visual C++, or Borland's Turbo C++ offer standard database operations within the programming language, including browse tables and screen formatting in Microsoft Windows-compatible modes. These languages offer a reasonable alternative to database packages for system development. However, it will be found to be significantly more difficult to use such systems than to use a standard package.

For users approaching the purchase of a package for the first time and requiring guidance, the author would recommend Microsoft FoxPro. It is widely used, offers high performance, there are many books and specialised training courses available for it, and it is supported on MS-DOS, Windows, and Unix systems. In benchmark tests, FoxPro 2.5 has been found to run faster than compiled Quick BASIC 4.5 in numerical and simulation applications.

The data analysis examples in this book are written in XBASE using statements and commands that are compatible with FoxPro 2.5. The use of DBF-structured data files is assumed throughout. Such files can be transferred easily between all major database, spreadsheet, and statistical analysis packages.

2.2.2 *Organization of a relational database*

Any forestry institution that is processing PSP data will need to maintain a number of different but related database files. These are likely to include:

- PSP tree data
- PSP regeneration data
- PSP descriptive and treatment data
- Experiment description and treatment data
- Experiment tree record data
- Species lists
- Ecological/seed characteristics of species

and so on.

It is important that every file should have a key field that will link its records with those in another file by a direct process of indexing. This can be illustrated, for example by considering three types of file that may be commonly encountered: The species list, PSP tree data, and PSP description data. For an elementary species list, we may have a file structure of the type shown in Example 1. In this the key field is a 4-digit code called SPP.

Field	**Type**	**Length**	**Usage**
SPP	N	4	Species code. This is the key field
SPNAME	C	40	Scientific species name
LNAME	C	40	Local name(s)
FAMILY	C	25	Botanical family

Example 1 *A simple species list file structure*

Field	**Type**	**Length**	**Usage**
PLOTID	N	4	Plot identification number
AYR	D	8	Date of assessment
QNO	N	2	Quadrat number
TNO	N	4	Tree number
SPP	N	4	Species code
LOCE	N	4.1	Tree location East
LOCN	N	4.1	Tree location North
DIAM	N	5.1	Diameter measurement
ADIAM	N	5.1	Alternate diameter at new POM
POM	N	5.2	Point of measurement
CFS	N	1	Crown Freedom Score
NOTES	C	20	Codes notes, etc.
CHKSUM	N	6.1	Checksum

Example 2 *A simple PSP tree record structure*

Example 2 shows a simple format for PSP tree data. The species code field SPP corresponds in type and length to the SPP field in the species list file. A second key field is also present, the PLOTID. This is also a numeric, 4-digit code, which serves two important purposes:

- It links all trees for the same plot.
- It links the tree records to plot-level information in another file.

The plot-level information is shown in Example 3. PSPs may typically be either simple monitoring plots, or assessment plots within an experimental design (Alder & Synnott, 1992). The record structure shown allows a narrative field called DIARY of unlimited length for each plot, a treatment code which is of importance for statistical analysis as a categorical variable, and other possible management and site information. The geographical coordinates are of some importance, allowing plot data to be linked to a Geographical Information System (GIS) such as ARC/INFO and plotted in map form. They are shown here in UTM format, but could equally well be given as longitude and latitude.

Field	Type	Length	Usage
PLOTID	N	4	Unique PSP number
EXPNO	N	3	Experiment number
TREATMENT	C	12	Treatment code
SITECODE	C	20	Comma separated 2-letter site codes
UTM_N	N	10	Geographical coordinate, UTM North
UTM_E	N	10	- " -, UTM East
YREST	N	4	Year PSP established
YRLOG	N	4	Year last logged
DIARY	M	10	A narrative recording any special influences on the plot. This is a Memo field of unlimited length.

Example 3 *Example of possible PSP heading file*

The key fields SPP and PLOTID maintain links between these files, as shown in the figure. During data entry, the SPP code can be looked up by the system in the species list to check its validity, and possibly display a botanical or local name to help avoid errors. When listing or analysing data, as will be shown in later examples, the SPP code is used to derive a GENSPP field in output data files that is important for analysing groupings of and differences between species.

Similarly, the PLOTID key groups together all the trees of a plot. This allows plot level parameters such as stem numbers, basal area, and so on to be calculated and output together with plot treatment code, geographical coordinates or other information extracted from the PSP heading file (Example 3).

These points are illustrated for a very typical situation with sample plot data in Figure 3. The PLOTID code links one record of plot-level information with many records of tree level information (a one-to-many relation in database parlance). For each tree record, the SPP code can be used to look up details of species name or other details.

Some general points may be made here about species codes and plot identification numbers.

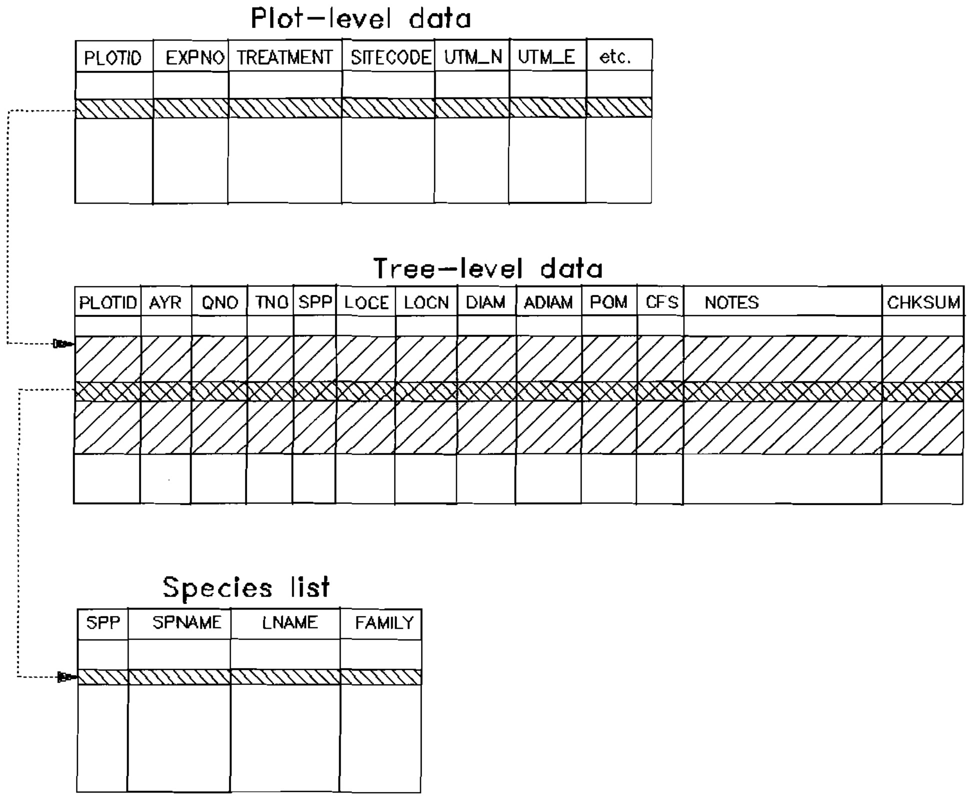


Figure 3 ***Linkages between related files in a PSP database***

(i) It is better to avoid complex hierarchical codes as key fields for linking database records. For example, one may design a species code number that incorporates order, family, genus, species numbers in a format like 01.23.431.2. Such numbering systems take up more space and are slower to key in when entering large masses of data, but most importantly, they are subject to change when the species is reclassified. This is a severe disadvantage. It is also more difficult to devise the validation and verification routines for such quasi-numeric codes.

Similarly, hierarchical codes are often devised for plot numbers, incorporating numbers, for example, for region, district, forest and plot. This is subject to the same disadvantages as noted above.

(ii) It is recommended that species codes and plot codes should be devoid of implicit meaning, simply arbitrary numbers used for locating fields in the database. For plot numbers, they should be unique at the national level. The numbers will then be brief, of simple structure, and invariant.

(iii) In some contexts, character fields are more appropriate. For example when creating files for analysis, a brief code that gives a mnemonic for a treatment or species can be used to label outputs and graphs more clearly than a numeric code. It is a simple matter to look up such mnemonic codes when they are required and substitute them for the original numeric codes.

Having noted that for new projects, simple numeric keys are easier to use, it must be accepted that in some cases, hierarchical numbering systems or the use of character or mnemonic codes may be established standards. In such cases, the database designer is obliged to accomodate them.

2.2.3 *Data file organization, nomenclature and archival*

It is suggested that the original data for each measurement period on a group of plots is maintained in a separate file, with file names being chosen to reflect the geographical unit (forest reserve, district, etc.), the measurement year, and the type of file (plot header, tree data, etc.). The file extension in MS-DOS will be determined by the database package, and may not be varied by the user. For example, all XBASE files will have extensions .DBF.

For example, the first letter of an 8-letter name may indicate the type of file (T for tree data, P for plot headers, R for regeneration strips, etc.); the next 4 letters may be reserved for the geographical unit (eg. forest reserve) and then two digits for the measurement year. This would give names such as:

PASUK98.DBF
TTAIN92.DBF

These files should be stored in a directory on disk reserved for PSP raw data. Consolidated files and others produced during analysis should be kept separately. The raw data directory can be called, for example, simply PSPS.

The plots grouped into a file should be restricted to keep file sizes to less than about 2 megabytes each, and preferably smaller.

Data files should be saved to diskette (1.4 MB high density format in DOS) using data compression utilities such as PKARC or the facilities provided with packages such as XTREE Gold. Such compression utilities are very effective with XBASE files and will usually achieve a size that is 20-30% of the uncompressed file.

At the same time, paper copies of data should be kept in files. It is suggested that one file is maintained per plot, which can be annotated as corrections are made, and may also record field work, forest operations, and other matters of relevance.

The archive diskettes should be stored in a fire-proof safe and reviewed and tested annually to avoid loss of data as a result of deterioration of diskettes. Standard utilities to test disks can be used, such as the Norton Disk Doctor program; or the file may be simply read in and decompressed.

Working copies of the data should be maintained elsewhere, either on diskette, on hard disk, or on back-up tapes. The archive disks should normally be kept locked away from casual access. At all times there should be two copies of the original data files, one of which is kept under secure conditions.

2.3 **Data entry and editing**

2.3.1 *Setting up a PSP data entry system with XBASE*

The simplest method of entering data in XBASE is to use the implicit BROWSE command. If for example the PSP data is stored in a database called PSP_T using the structure shown in Example 2, then the sequence of commands shown in Example 4 will allow data entry and editing of a single assessment. In this example, the file name suffix _T is intended to indicate that the file contains tree-level information. However, the file name can be made up in any way that suits the user.

```
* Index file in plot-quadrat-tree-year order.  The index is called PQTY.
* This command is only needed at the first editing session
INDEX ON STR(PLOTID,4)+STR(QNO,2)+STR(TNO,4)+DTOS(AYR) TAG PQTY

* Open the data editing window with the PQTY index active, and automatic
* carry down of Plot number, year, and quadrat number.
SET ORDER TO PQTY
SET CARRY TO PLOTID, AYR, QNO
SET CONFIRM ON
SET BELL OFF
BROWSE COMPRESS NOMENU
```

Example 4 *Commands for PSP data entry using a simple BROWSE*

The SET ORDER command in the example ensures existing records are presented sorted by an index tag called PQTY. At the bottom of the example, the INDEX command that defines PQTY is shown. It sorts records by plot number, quadrat number, tree number and assessment year, so that records for the same tree made in successive years will be adjacent. The SET CARRY command ensures that data which is mainly common to successive entries is copied down automatically. This comprises plot and quadrat numbers and the assessment date. These carried-down entries can be altered if required by editing the appropriate field on the record being entered, as for example when moving to the next quadrat or plot.

The SET CONFIRM and SET BELL commands are matters of choice. The former requires that the **Enter** key is pressed before moving to the next field, whilst the latter switches off the audible tone each time a field is filled. The user may experiment with setting these options ON or OFF according to preference. The NOMENU and COMPRESS options on the BROWSE command are also optional. NOMENU restricts the actions the data entry personnel can perform to simple entry and editing of data. The menu, if present, allows the operator to reindex the file and manipulate it in other ways that may not be desirable. The COMPRESS option gives a slightly neater presentation of the browse screen that allows more data lines to be seen.

The INDEX command shown in the box need in principle be issued only once, when the file is created. It converts the various numeric and date fields into character form with the STR() and DTOS() functions and joins them into a single indexing variable that will ensure records are presented in an appropriate order. This index will be maintained automatically by XBASE under as new records are added. However, if there is a power failure or other system shut down whilst the data file is open, the index may be corrupted. In this case, the INDEX command shown can be be issued to recreate it.

It will be noted that the indexing command will be simplified if PLOTID, QNO, TNO are declared as character fields. However, additional problems then arise relating to:

- The different collating sequences that are used for character and numeric values.
- Entry of variable numbers of leading blanks give different values when character fields are used.

In the author's experience, it is generally better to use numeric-type fields where numeric values are involved, in spite of the more complicated expressions required for indexing. For example, if the PLOTID is a 4-character field, then the entries:

```
'52  '
' 52 '
'  52'
```

would all be distinct and unequal text values. With numeric fields, no such ambiguity exists.

The statements in Example 4 can be edited into a small command file using the MODIFY COMMAND statement, and then run with the DO command, avoiding the need to re-enter all the statements each time editing is done.

2.3.2 *Testing Checksums*

The author has recommended elsewhere (Alder & Synnott, 1992, page 115) that a checksum field is added to each data record to help check for correct data entry. A checksum field is also shown in Example 2.

The checksum field simply contains the total of all the numeric fields in each record, and is entered manually on the field forms prior to data entry.

Ideally, the checksum would be tested by the computer as each line of data is entered. With XBASE this is difficult to arrange and requires considerable programming.* However, it is comparatively easy to write a generic checksum

* The dbase VALID clause on a GET statement, combined with a user-defined function, might seem to offer a simple route to this end. It does not however work in practice as values entered into fields during GET processing are not updated in the program until after the record is completed. The CHECKSUM can therefore only be verified after moving to a new record, and not immediately after leaving the CHECKSUM field, which is the facility automatically available with the VALID clause.

```
* program to do checksums and edit records in error
* all numeric fields are used to calculate checksums
set talk off
clear
* count fields in database
nf=1
do while len(field(nf))>0
  nf=nf+1
enddo
nf=nf-1
declare arec[nf]
scan
  * calculate checksum and compare with entered value
  csum=0
  i=1
  copy to array arec record recno()
  do while i<nf
    if type("arec[i]")="N"
      csum=csum+arec[i]
    endif
    i=i+1
  enddo
  if csum< >arec[nf]
    * checksum error : beep and enter browse screen
    ? chr(7)
    set message to "Check sum error : Record "+str(recno(),6)
    * remember record number (in case user moves off record inside browse)
    nrec=recno()
    browse compress nomenu
    if lastkey()=27
      * cancel program if ESC pressed
      set message to "Check sum processing cancelled by Esc"
      exit
    endif
    goto nrec
  else
    set message to "Check sums OK up to record "+ str(recno(),6)
    skip
  endif
endscan
set talk off
return
```

Example 5 *Program CHECKSUM for checksum verification and data editing*

validation program that works on the file in batch mode. The program shown in Example 5 will work on any XBASE file that uses the last field as a checksum. It totals all numeric fields and compares them with the CHECKSUM field. When an error is found it enters BROWSE mode to allow correction. Checking continues until the file is clean. To operate the program, the following commands may be used within XBASE:

```
USE PSPDATA
DO CHECKSUM
```

This assumes that the data file to be checked is in a file called PSPDATA.DBF, and the program is in a file called CHECKSUM.PRG. Once an erroneous record has been corrected within the BROWSE editor, checking is resumed by pressing **Ctrl-End**. The **Esc** key will terminate the program without completing checks.

2.3.3 *Range and validity checks*

Although checksums are a powerful tool, they are not always appropriate. In particular, if a field-portable computer is used to directly key in data as it is measured, there is no role for checksums. They are essentially a device for verification of key-punched data from hand-written forms.

Editing program PSPEDIT.PRG

```
procedure pspedit
parameters pspfile
use (pspfile) order PQTY alias pf in 1
use species order SPP alias sp in 2
use notelist order CODE alias nt in 3
select 1
set format to psp
set procedure to pspvalid
set confirm on
set carry to PLOTID, AYR, QNO
browse nomenu compress format
return
```

Format file PSP.FMT

```
@ 1,0 get PLOTID picture [9999]
@ 2,0 get AYR    picture [99-99-99]
@ 3,0 get QNO    picture [99]
@ 4,0 get TNO    picture [9999]
@ 5,0 get SPP    picture [9999] valid seek(pf->SPP,"sp") error "No such species code"
@ 6,0 get LOCE   picture [99.9] range 0,20
@ 7,0 get LOCN   picture [99.9] range 0,20
@ 8,0 get DIAM   picture [999.9] range 10,300
@ 9,0 get ADIAM  picture [999.9] range 10,300
@ 10,0 get POM   picture [99.99]
@ 11,0 get CFS   picture [9] range 1,5
@ 12,0 get NOTES picture [XX,XX,XX,XX,XX,XX,XX] valid notechek(NOTES) ;
             error "Invalid note code"
@ 13,0 get CHKSUM picture [9999.9]
```

Example 6 *Method of setting up range and validation checks for the DBASE browse editor*

Range and validity checks can detect some errors that may arise at the measurement or original data recording stage. Most database packages provide some method for simple error checking. In XBASE, if data entry is done via the BROWSE command, then a *format file* must be defined that includes range and validity functions. The validation functions can themselves be user-written programs contained in a *procedure file*. In Example 6, the main editing file is shown as PSPEDIT.PRG. This sets up the format and procedure files, and starts the browse editor. The format file is called PSP.FMT. Two validation procedures are called : **seek** and **notechek.** The former is an intrinsic XBASE function, whilst the latter is contained in the procedure file declared in the SET PROCEDURE TO statement, and is shown Example 7. These examples assume the use of a PSP database with the structure shown on page 12.

```
function notechek
* checks a list of notes like 'DT,BS,FD,CD, , , ' to make sure each 2-
* letter pair can be found as a CODE in file NOTELIST. If it can't, a
* pop-up box of possible notes is offered. <Return> selects an entry and
* makes a correction; <Esc> returns to the calling validation program with
* a value FALSE.
parameters tnotes
select nt
nc=1
ok=.T.
do while nc<len(tnotes)
  set order to tag CODE
  seek substr(tnotes,nc,2)
  if .not. found()
    define popup choices from 4,5 to 19,45 prompt field NOTE_TEXT
    on selection popup choices deactivate popup
    activate popup choices
    if lastkey()=27
      * no choice made : <Esc> key pressed. Return with value FALSE
      ok=.F.
      exit
    else
      * find choice code and substitute in string
      set order to tag NOTE_TEXT
      seek prompt()
      tnotes=stuff(tnotes,nc,2,CODE)
      select pf
      replace NOTES with tnotes
      select nt
    endif
  endif
  nc=nc+3
enddo
select pf
return ok
```

Example 7 *Validation procedure NOTECHEK to check coded notes and provide pop-up help box*

The range tests included are:

- The DIAM and ADIAM field must be in the range 10-300 cm.
- The LOCE and LOCN fields must be from 0-20 m.

- The CFS fields must all be 1-5.

The validation functions include:

- A check that the species code exists by looking it up in file SPECIES.DBF with the intrinsic **seek** function.

- A check that all the note codes entered can be found in the database file NOTELIST.

In the format file, the coordinates used on the GET commands are arbitrary and are ignored by XBASE. They must however be entered in order to complete the syntax of the command. Standard XBASE documentation gives more information on this point.

The PSPEDIT program in this example is started within XBASE by typing:

```
DO PSPEDIT WITH pspfile
```

where 'pspfile' is the name of the data file being entered.

Procedures of this type can be set up in field-portable computers that are MS-DOS compatible. An XBASE compiler such as CLIPPER is useful in such contexts to produce compiled stand-alone editing and entry programs that can be downloaded onto the field computer without requiring the disk storage and memory overheads of a complete interactive database package.

2.3.4 *Merging and checking successive measurements*

When a database package is used to manage PSP data, the merging of subsequent measurements with earlier ones becomes a trivial task. Programming is minimized if the re-measurement data is initially entered into an independent file. This file is then merged with earlier measurements to form a consolidated data set.

Assuming that the suggestions in section 2.2.2 have been followed, and there are two files in use:

PSPCD	Consolidated PSP data for a series of preceding years.
PSP95	PSP data for the 1995 remeasurement.

The PSP95 measurement data can be appended and correctly merged with the earlier measurements with the following XBASE commands:

```
USE PSPCD ORDER TAG PQTY
APPEND FROM PSP95
```

The appended records will appear in their correct place with preceding measurements for the same tree whilst the PQTY (Plot Quadrat Tree Year) index is active.

Physically, the appended PSP95 records will be at the end of the PSPCD file and will appear there if no index is active.

```
* checks merged PSP data for (1) changes in species code, (2) diameter
* increments outside range -0.1 to +3.0 cm/year.
set talk off
use pspcd order tag pqty in 1
treeid0='?'
* examine data records in sequence
scan
  * ID of this tree
  treeid1=str(PLOTID,4)+str(QNO,2)+str(TNO,4)
  * see if same as preceding line of data
  if treeid1=treeid0
    * compare species, diam with last measurement.
    nrec=recno()
    nyr=(AYR-year0)/365
    dinc=(DIAM-diam0)/nyr
    do case
    case SPP< >spp0
      set message to "Change in species code"
      skip -1
      browse nomenu compress
    case dinc<-0.1 .or. dinc>3
      set message to "Unlikely diameter measurement"
      skip -1
      browse nomenu compress
    endcase
    if lastkey()=27
      * <esc> in browse terminates program
      exit
    endif
    * re-position file to record current when entering browse
    goto nrec
    set message to
  endif
  * update treeid, species, diam for a new tree
  treeid0=treeid1
  spp0=SPP
  diam0=DIAM
  year0=AYR
endscan
set talk on
set message to
return
```

Example 8 *Program SEQCHEK to check for logical errors in sequential tree measurement records*

The consolidated data file can then be checked in batch mode for illogical changes in sequential measurements. For example, changes in species code, or reductions in diameter can be detected. Program SEQCHEK in Example 8 will scan a sequence of measurements for the same tree and pause in BROWSE mode to allow data examination and correction if either the species code changes or diameter shrinks by more than 1 mm/yr*. The program is set up to operate on a file of consolidated measurements called PSPCD, with the structure shown on page 12.

This program illustrates two useful and common techniques when processing sequential measurement data, with one record per measurement:

- Changes in tree identification are used to flag the start and end of a measurement series for one tree. The value TREEID0 holds the ID of the tree referred to in a preceding record; TREEID1 is calculated for the current record. If they differ, then the sequence relates to a new tree; if they are the same, then calculations and comparisons can be made between this record and the last.

- Increment is calculated by retaining the diameter and measurement date for the preceding measurement of a tree in DIAM0 and YEAR0. Increment is then derived as:

  ```
  DINC=(DIAM-DIAM0)/((AYR-YEAR0)/365)
  ```

  where DIAM is the diameter field on the current record, and AYR is the measurement date. In XBASE, substraction of date values gives the time difference in days; division by 365 gives a time difference in years.

Whilst merged records can be maintained in their correct sequence in XBASE by using an appropriate index, this is not possible if the file is read by another package such as SYSTAT. It is suggested that the consolidated file with merged records is copied, under index control, to a second file for long-term storage and use by other programs. The sequence of commands:

```
USE PSPCD ORDER TAG PQTY
SET DELETED ON
COPY TO PSPCDS
```

will create a second database file PSPCDS.DBF identical to the first, but without records marked for deletion in PSPCD, and with the records physically sorted in the correct sequence.

2.3.5 *Database entry formats using multiple years per record*

The record formats so far discussed use one record per year of measurement. This may be called a year-per-record (YPR) format. An alternative approach which is suitable in some circumstances is a format in which the various tree observations (dbh, crown position, etc.) are entered in columns across the record, with one

* Small shrinkages between measurements may occur and should be allowed for in data checking routines. Shrinkage may result from moisture stress on trees with little or no growth; or as a result of instrument error. Artificially eliminating all negative increments biases the increment error distribution and should be avoided. However large negative increments suggest recording or procedural errors that should be corrected.

column per year. The data for several years will thus be contained on a single database record. Such a format may be termed a multi-year record (MYR) format.

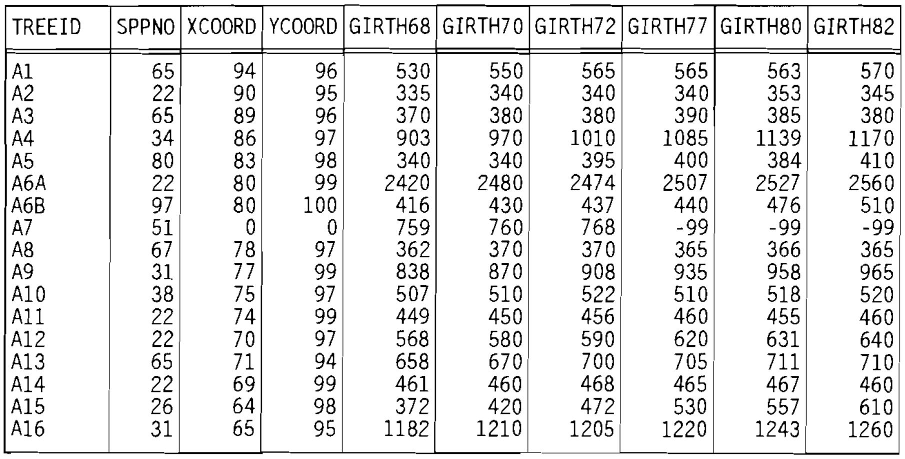
An example is illustrated in Figure 4, which shows part of a file of data from PSPs at Kade, Ghana (Swaine et al., 1987) as a XBASE browse screen. In the format used, the time-invariant fields are at the left, and include tree identification, species code, and tree coordinates. Then follow a series of columns for each year of measurement, containing in this case girths in mm. GIRTH82, for example, indicates girths measured in 1982. Negative values are used as codes, with -1 indicating undersized trees (below 300 mm girth) and -99 dead trees. To add a new year of measurement to such data, the structure is modified by adding a new field to the end of the record structure.

In a case such as this, where only a single variable is measured, this is a quite convenient format. Even when 2 or 3 variables a measured in each year, the MYR format may be used, according to the personal preferences of the system designer. However, with more variables it becomes rather cumbersome, as records become very long. Even with a single variable, listing of MYR format data with many years of measurement may exceed the paper width of most printers.

Many statistical packages cannot directly plot or analyse the data in this format, and may require it to be converted into a YPR format. Generally, it is the author's experience that:

(i) The YPR format is more flexible and convenient to use when setting up new systems.

(ii) For coding existing data sets which have been previously set out in ASCII files or measurement ledgers in a year-per-column arrangement, the MYR format may give the simplest conversion or data entry into XBASE, and may therefore be used initially. Subsequently, programs must be written to convert MYR into YPR format for analysis by statistical packages such as SYSTAT.



TREEID	SPPNO	XCOORD	YCOORD	GIRTH68	GIRTH70	GIRTH72	GIRTH77	GIRTH80	GIRTH82
A1	65	94	96	530	550	565	565	563	570
A2	22	90	95	335	340	340	340	353	345
A3	65	89	96	370	380	380	390	385	380
A4	34	86	97	903	970	1010	1085	1139	1170
A5	80	83	98	340	340	395	400	384	410
A6A	22	80	99	2420	2480	2474	2507	2527	2560
A6B	97	80	100	416	430	437	440	476	510
A7	51	0	0	759	760	768	-99	-99	-99
A8	67	78	97	362	370	370	365	366	365
A9	31	77	99	838	870	908	935	958	965
A10	38	75	97	507	510	522	510	518	520
A11	22	74	99	449	450	456	460	455	460
A12	22	70	97	568	580	590	620	631	640
A13	65	71	94	658	670	700	705	711	710
A14	22	69	99	461	460	468	465	467	460
A15	26	64	98	372	420	472	530	557	610
A16	31	65	95	1182	1210	1205	1220	1243	1260

Figure 4 *BROWSE screen for data in a multi-year record format*

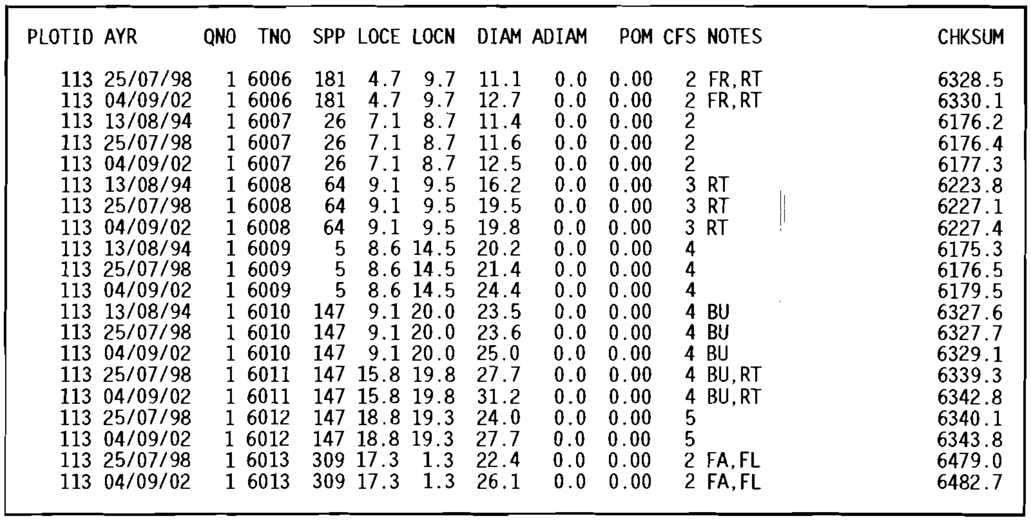
2.4 Printing and mapping PSP tree data

2.4.1 *Reference listings of plot data*

Listings of plot data are useful for error checking, for maintaining hardcopy documentation on plots, and for other purposes. Generally listings can be divided into two categories:

- Those presenting the information precisely as it is stored in the database.
- Those with additional look-up or calculated values giving, for example, species names instead of codes.

In addition summary listings may be required for control and work management purposes. For example, a simple listing of plot numbers with numbers of trees may be made to show work entered into the computer.



```
PLOTID AYR       QNO  TNO  SPP LOCE LOCN  DIAM ADIAM   POM CFS NOTES                      CHKSUM

   113 25/07/98    1 6006  181  4.7  9.7  11.1   0.0  0.00   2 FR,RT                      6328.5
   113 04/09/02    1 6006  181  4.7  9.7  12.7   0.0  0.00   2 FR,RT                      6330.1
   113 13/08/94    1 6007   26  7.1  8.7  11.4   0.0  0.00   2                            6176.2
   113 25/07/98    1 6007   26  7.1  8.7  11.6   0.0  0.00   2                            6176.4
   113 04/09/02    1 6007   26  7.1  8.7  12.5   0.0  0.00   2                            6177.3
   113 13/08/94    1 6008   64  9.1  9.5  16.2   0.0  0.00   3 RT                         6223.8
   113 25/07/98    1 6008   64  9.1  9.5  19.5   0.0  0.00   3 RT                         6227.1
   113 04/09/02    1 6008   64  9.1  9.5  19.8   0.0  0.00   3 RT                         6227.4
   113 13/08/94    1 6009    5  8.6 14.5  20.2   0.0  0.00   4                            6175.3
   113 25/07/98    1 6009    5  8.6 14.5  21.4   0.0  0.00   4                            6176.5
   113 04/09/02    1 6009    5  8.6 14.5  24.4   0.0  0.00   4                            6179.5
   113 13/08/94    1 6010  147  9.1 20.0  23.5   0.0  0.00   4 BU                         6327.6
   113 25/07/98    1 6010  147  9.1 20.0  23.6   0.0  0.00   4 BU                         6327.7
   113 04/09/02    1 6010  147  9.1 20.0  25.0   0.0  0.00   4 BU                         6329.1
   113 25/07/98    1 6011  147 15.8 19.8  27.7   0.0  0.00   4 BU,RT                      6339.3
   113 04/09/02    1 6011  147 15.8 19.8  31.2   0.0  0.00   4 BU,RT                      6342.8
   113 25/07/98    1 6012  147 18.8 19.3  24.0   0.0  0.00   5                            6340.1
   113 04/09/02    1 6012  147 18.8 19.3  27.7   0.0  0.00   5                            6343.8
   113 25/07/98    1 6013  309 17.3  1.3  22.4   0.0  0.00   2 FA,FL                      6479.0
   113 04/09/02    1 6013  309 17.3  1.3  26.1   0.0  0.00   2 FA,FL                      6482.7
```

Example 9 ***Listing of PSP data produced by simple DBASE LIST command***

All these listings can be produced easily using standard database report generators or simple commands. In XBASE, LIST can be used to produce raw data output, and may also include lookup values such as species names.

For example, with the PSPCD database, the commands:

```
USE PSPCD ORDER TAG PQTY
LIST OFF NEXT 20
```

will list the first 20 lines in the file, as shown in Example 9. The OFF keyword switches off the listing of record numbers, allowing more columns of data to fit across the paper. A complete plot, for example No. 113, could be listed by:

```
SEEK ' 113'
LIST OFF WHILE PLOTID=113
```

Note in this example that the PQTY index is built as a character string (see INDEX command in Example 4), whilst PLOTID is a numeric field. Hence 113 is entered as a right-justified 4-character string on the SEEK command to find the first record in the plot, but as a numeric value (without quotes) on the WHILE clause. The latter will cause the listing to continue until the first record encountered with a PLOTID value that is not 113.

Commands input

```
use pspcd order tag pqty alias p in 1
use species order tag spp alias s in 2
set relation to spp into s
list off fields qno,tno,spp,s->loc_name,diam for year(ayr)=1998
```

Listed output

```
qno  tno   spp s->loc_name                 diam
  1 6000   147 Laurel                      14.9
  1 6001   147 Laurel                      15.9
  1 6002   147 Laurel                      45.9
  1 6003    56 White Breadnut              16.0
  1 6004   101 Botan palm                  26.8
  1 6005   181 Copalche                    17.6
  1 6006   181 Copalche                    11.1
  1 6007    26 Wild Pear (Aguacatillo)     11.6
  1 6008    64 Cherry                      19.5
  1 6009     5 Moho                        21.4
  1 6010   147 Laurel                      23.6
  1 6011   147 Laurel                      27.7
  1 6012   147 Laurel                      24.0
  1 6013   309                             22.4
  1 6014    13 Hogplum                     32.1
  1 6100    45 Sillion                     33.1
  1 6101    25 Cedrillo                    48.6
  1 6102     5 Moho                        20.9
```

Example 10 *Listing PSP data using relational linkage to show species names*

It is useful for field crews to have an abbreviated listing of the data at the last measurement, including tree number, species and diameter to the nearest cm, to help them locate and check measurements as they are made. In such listings, the species codes can conveniently be replaced by vernacular or botanical names. Example 10 shows a group of instructions, assuming that species codes are in a database SPECIES. Both the PSP data file and the species file are opened before the list command is given; the SET RELATION command establishes a linkage using the SPP field, which must be the active index in the species file. The LOC_NAME field in the species file contains the local names for each species code, which are reproduced on the listing as shown.

2.4.2 *Producing plot maps*

Plot maps are another useful technique for documenting data and assisting field work. They are comparatively easy to produce directly from XBASE or a similar text-based database package using the industry-standard HPGL[*] plotting instructions.

[*] Hewlett-Packard Graphics Language. HPGL is standard with almost all graph plotters, and is recognised by many word processing packages such as Word Perfect. HPGL files can also be plotted directly by some laser printers. Appendix A documents HPGL instructions used in this manual.

Before plotting a tree map, a database file needs to be set up showing the coordinate positions in metres of the south-eastern corner of each quadrat relative to the south-eastern corner of the plot. Different organizations will use different quadrat numbering schemes, but generally only one such reference file will be needed for a large set of plots. Figure 5 shows a typical scheme (after that shown in Alder & Synnott, 1992, page 60) together with coordinates to be saved in a file called QUADCORD.

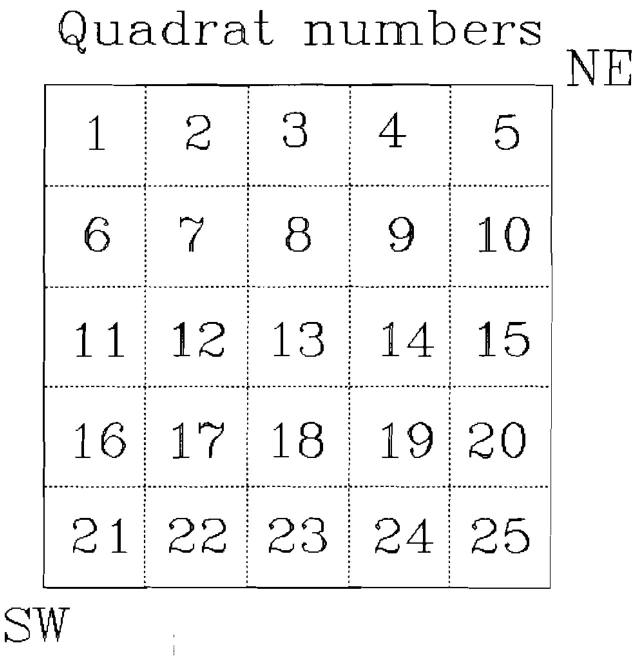
Program PSPMAP.PRG, listed in Example 11, scans a data file for a single PSP and produces a map such as that shown in Figure 6. The plot is produced as a text file PSPMAP.PLT containing the HPGL instructions; the first few lines of this output file are reproduced in Example 12. The HPGL file can be sent directly to a physical device capable of interpreting them, such as a graph plotter, using the DOS COPY instruction, as for example:

```
COPY PSPMAP.PLT COM1:
```

Alternatively, the file can be read in to a package that can manipulate it as graphics and print it on a standard printer. In this book, HPGL files have been imported directly into Word Perfect to reproduce the figures.

The program works in the following stages:

- The scale and origin of the plot are controlled by the variables X0, Y0 and



Quadrat SE corner coordinates

QNO	LOCE	LOCN
1	0	80
2	20	80
3	40	80
4	60	80
5	80	80
6	0	60
7	20	60
8	40	60
9	60	60
10	80	60
11	0	40
12	20	40
13	40	40
14	60	40
15	80	40
16	0	20
17	20	20
18	40	20
19	60	20
20	80	20
21	0	0
22	20	0
23	40	0
24	60	0
25	80	0

Figure 5 *A quadrat numbering scheme with database entries for corresponding quadrat coordinates*

```
procedure pspmap
* program to generate map in HPGL of a permanent sample plot
parameter pspfile
set talk off
* PSP origin (SE corner in physical plotter units) and plotter units/metre
x0=1000
y0=1000
pum=60
set printer to file (pspfile+".PLT")
set printer on
* initialize plotter, select pen 1, character terminator ~, char size
? "IN; SP 1; DT~; SR 0.5,1 ;"
* open PSP data file, link to quadrat coordinate file
select 1
use (pspfile) order tag PQTY alias p in 1
use QUADCORD order tag QNO alias q in 2
set relation to QNO into q
* draw plot frame
? "PU "+str(x0,8)+str(y0,8)+";"
? "PD "+str(x0+100*pum,8)+str(y0,8)+";"
? "PD "+str(x0+100*pum,8)+str(y0+100*pum,8)+";"
? "PD "+str(x0,8)+str(y0+100*pum,8)+";"
? "PD "+str(x0,8)+str(y0,8)+";"
nq=1
* draw internal plot frame with dashed lines then revert to solid lines
? "LT 2,1;"
do while nq<5
  ? "PU "+str(x0+20*nq*pum,8)+str(y0,8)+";"
  ? "PD "+str(x0+20*nq*pum,8)+str(y0+100*pum,8)+";"
  ? "PU "+str(x0,8)+str(y0+20*nq*pum,8)+";"
  ? "PD "+str(x0+100*pum,8)+str(y0+20*nq*pum,8)+";"
  nq=nq+1
enddo
? "LT;"
* scan data file tree by tree
select p
scan
  * move pen to tree position, write tree number, draw a circle
  lx=LOCE+q->LOCE
  ly=LOCN+q->LOCN
  ? "PU "+str(x0+lx*pum,8)+str(y0+ly*pum,8)+";"
  ? "LB"+ltrim(str(TNO,4))+"~ "
  ? "PU "+str(x0+lx*pum,8)+str(y0+ly*pum,8)+";"
  ? "CI "+str(DIAM/100*2*pum,8)+";"
endscan
? "SP;"
set printer off
set printer to
set talk on
return
```

Example 11 *Program PSPMAP to draw a map of trees on a PSP using HPGL commands from XBASE*

PUM set at lines 6-8. X0 and Y0 set the origin of the bottom left (south-east) corner of the plot in physical coordinates on the paper. The PUM variable gives the number of physical plotter units per metre on the ground. The value given is appropriate for A4 paper and a 100 m square plot.

- Printer output from the XBASE "?" command is redirected to a file in lines 9-10. This redirection is switched off at lines 47-48 and the output file closed. If the database file was, for example PSP113, the the HPGL output file would be called PSP113.PLT.

- The plotter is initialized at line 12. The character ~ is defined as a string terminator for the HPGL "LB" command, and character width and height are set to 0.5% and 1% of paper width and height respectively.

- Lines 14-17 open the input file and link it to the quadrat coordinate file using the quadrat number as a key field.

- Lines 19-23 draw the PSP boundary as a solid line. Annotation of the plot with the PSP number or other text could be included here but have been omitted in order to keep this example program as brief as possible.

```
IN; SP 1; DT~; SR 0.5,1 ;
PU      1000      1000;
PD      7000      1000;
PD      7000      7000;
PD      1000      7000;
PD      1000      1000;
LT 2,1;
PU      2200      1000;
PD      2200      7000;
PU      1000      2200;
PD      7000      2200;
PU      3400      1000;
PD      3400      7000;
PU      1000      3400;
PD      7000      3400;
PU      4600      1000;
PD      4600      7000;
PU      1000      4600;
PD      7000      4600;
PU      5800      1000;
PD      5800      7000;
PU      1000      5800;
PD      7000      5800;
LT;
PU      1390      6634;
LB2~
PU      1390      6634;
CI        40;
PU      1654      6082;
LB3~
PU      1654      6082;
CI        19;
PU      1984      6208;
LB4~
PU      1984      6208;
CI        27;
```

Example 12 *Part of HPGL output file for PSP map*

- Lines 24-34 draw the internal quadrat boundaries on the map as dashed lines. The LT command initially sets small dashes, and then at line 34 resets solid lines as the default.

- Lines 37-45 scan through the data file and plot each tree. The LB command writes the tree number on the map. The CI command draws a circle which is scaled here by a factor of 2 to give a clearer plot. It might be desirable to change pen colour for the tree numbers by preceding the LB command with "SP 2;", for example, to select pen number 2. Similarly the tree circles could be plotted in a third colour by preceding the "CI" command with "SP 3;".

- Lines 46-50 park the plotter pen, close the output file, and restore the XBASE interactive dialogue.

Data for a given sample plot is extracted from the main PSP file with instructions such as:

```
USE PSPCD ORDER TAG PQTY
SEEK ' 113'
COPY TO PSP113 WHILE PLOTID=113 FOR YEAR(AYR)=94
```

The above sequence would copy data for PSP No. 113 and measurement year 1994 to a file PSP113. The PSPMAP program would then be run with this data with the command:

```
DO PSPMAP WITH "PSP113"
```

The output HPGL file would be called PSP113.PLT. If output is desired directly to a plotter, then line 9 of the program could be amended to, for example:

```
SET PRINTER TO COM1
```

This would direct output to the DOS COM1: serial port.

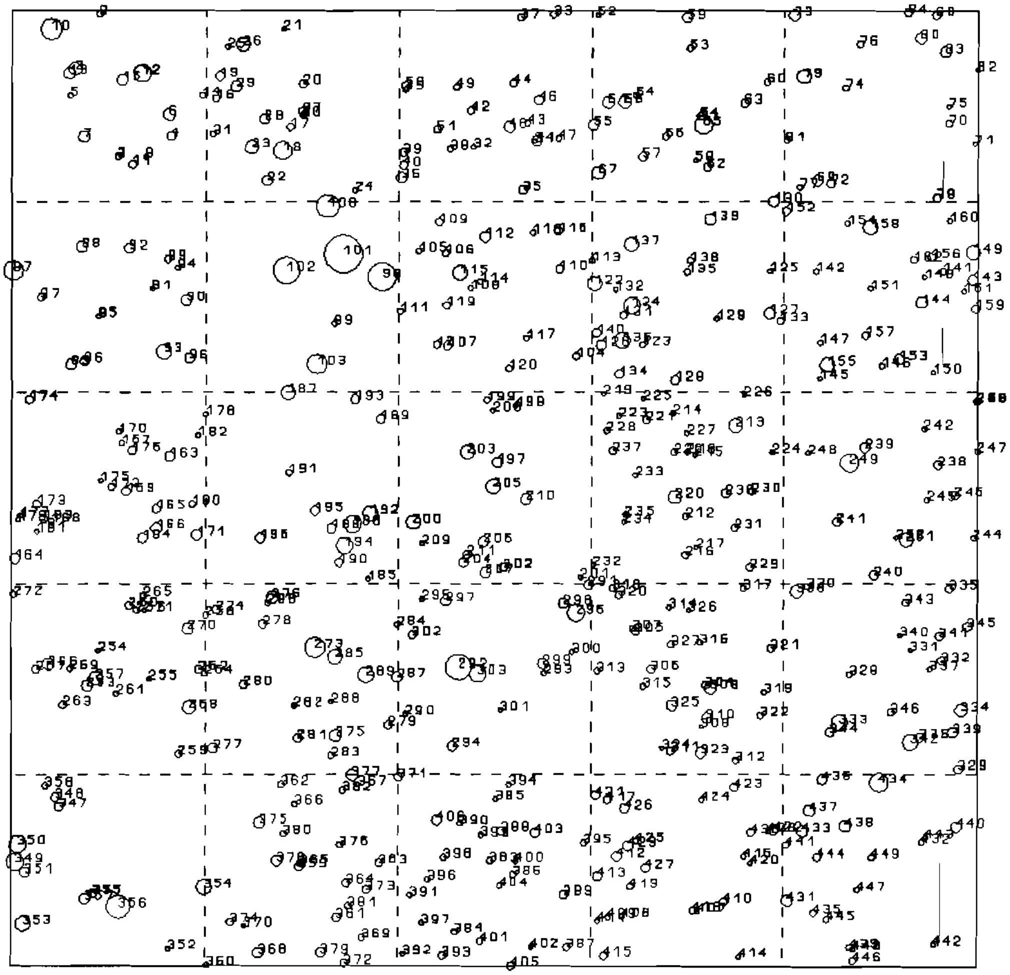


Figure 6 *PSP plot map showing tree numbers and relative sizes*

2.5 Preliminary operations for data analysis

2.5.1 *Standardization of measurement units*

PSP data may commonly be collected over long periods of time. Over such periods, methods of measurement and units of measure may change, making the raw data units incompatible. For example, one series of data the author worked with commenced in 1946 and included measurements up to 1992. During this 46-year period, diameter measurement was moved from 1.3 metres to 4 metres, and then back to 1.3 metres, units changed from inches girth to cm girth and then cm diameter, crown classification changed through three different systems, and tree numbers were completely altered three times (Alder, 1993).

Generally, conversions may be of two basic types:

- Continuous measures, such as diameter, which may be scaled or modifed by an arithmetic expression.

- Categorical variables, such as coded notes, species codes, crown classifications, which may be amended by the use of look-up tables or simple replacement for isolated changes.

Using XBASE, continuous measures may be adjusted by replacing a field value by an arithmetic expression based on its current value, for a selected set of records. For example, if a PSP database has been created with a field GIRTH containing girth measurements in inches prior to 1975, and a field DIAM containing diameters in cm from 1975, then the DIAM field for the pre-1975 records can be filled in with the expression:

```
REPLACE DIAM WITH GIRTH*2.54/3.14159 FOR YEAR(AYR)<1975
```

If a single code has to be replaced with an alternate value, then this can be done similarly. For example, the author found in one case that for a set of plots, the code "FR" had been used in place of the code "FD" for fire damage. The codes were stored in a string similar to the NOTES field shown in Example 9 on page 25.

The replacement was carried out with the following command:

```
REPLACE NOTES WITH STUFF(NOTES, AT("FR",NOTES),2,"FD") ;
        FOR "FR" $ NOTES
```

The FOR clause uses the $ operator to detect the string "FR" in the NOTES field. The STUFF() function replaces 2 characters starting at the position given by the AT() with "FD". The AT() function gives the character position of the "FR" string in NOTES.

A REPLACE command such as the above may not work reliably if the field being amended is the controlling index of the file; alteration of the index variable will lead to the sequence in which records are accessed being altered, so that some records may be skipped. If an index field is to be amended, the SET ORDER TO command should be used to select another key or process records in their natural order.

The more complex case is where a number of categorical codes have to be replaced. In this case, a second file should be set up which:

- Has a key field based on the old value to be changed.
- Gives the corresponding new value to be inserted.

For example, on some sample plots in Belize, the author found that two different species lists had been used for plots established at different periods, each with their own code numbers. It was necessary to consolidate the lists and assign new numbers that were standard on both groups of plots.

A database file was created called SPCHANGE with three fields, SPPA, SPPB and NEWSPP. The SPPA and SPPB fields contained numbers for a given species on the A and B series plots; the NEWSPP field was the corresponding new standard number to be applied.

To update the A series plots in a file called PSPA, the following commands were applied:

```
SELECT 1
USE PSPA
SELECT 2
USE SPCHANGE
INDEX ON SPPA TAG SPPA
SELECT 1
REPLACE SPP WITH SPCHANGE->NEWSPP FOR SEEK(SPP,"SPCHANGE")
```

The effect of these commands is to open the PSPA data file in work area 1 and the list of old and new code numbers in area 2. The SPCHANGE file is indexed on the SPPA codes. The replace command replaces the SPP number in the PSPA file with NEWSPP in the SPCHANGE file only if SPP corresponds to a value found in the SPCHANGE file in the SPPA field. The SEEK function will position the SPCHANGE file at the correct record to make the right substitution.

These examples indicate various techniques that may be applied for standardizing data and codes on to a common basis. It will be obvious that small errors in applying such powerful commands can have a catastrophic effect on the data, and it is therefore *essential to make a back-up copy of the original file before commencing work.*

2.5.2 *Correction for moving point of diameter measurement*

One special problem of data standardization that arises with measurements of tropical tree diameters is the occurrence of buttresses and the inevitable need to move the point of measurement (POM) of the reference diameter. Alder & Synnott (1992, p.71) suggest field procedures to deal with this problem. The tree record file whose structure is shown in Example 2 provides fields to record the necessary correction data. These are:

DIAM The diameter at the current POM.
ADIAM An alternate diameter measurement, made above the height at which DIAM is measured.
POM The height at which ADIAM is measured.

The way in which these fields might be filled in is illustrated for a sequence of measurements in which diameter is initially recorded at 1.3 m (the 'breast height' standard), then moved to 2.5 m, and moved again to 4 m. These are shown in Lotus 1-2-3 spreadsheet format in Example 13. In 1998, the tree is measured at the original point of measurement as 44.2 cm, and at the alternate diameter as 40.5 cm. At the next measurement, in 2000, the original diameter is no longer measured, and the alternate diameter now becomes the standard point of measurement. The ADIAM and POM fields are left empty in this and subsequent years until a new POM becomes necessary. In the example this is shown in the year 2006, when the POM is moved to 4 m. The diameter at 2.5 m is recorded as 48.1 cm, and at 4 m as 44.2 cm. Subsequent measurements are made at 4 m height.

	A	B	C	D	E	F
1	Year	Diam	Alt.	P.o.m.	Corr.	Ref.
2		(cm)	diam	(m)	ratio	diam
3	1994	41.7		1.30	1.000	41.7
4	1996	42.8			1.000	42.8
5	1998	44.2	40.4	2.50	1.000	44.2
6	2000	41.5			1.094	45.4
7	2002	43.3			1.094	47.4
8	2004	45.7			1.094	50.0
9	2006	48.1	44.2	4.00	1.094	52.6
10	2008	45.1			1.191	53.7
11	2010	46.2			1.191	55.0
12	2012	47.1			1.191	56.1

+B12*E12

@IF(C11>0,E11*B11/C11,E11)

Example 13 *Spreadsheet to calculate diameter corrections for a moving POM*

The spreadsheet and database presentations of the data are equivalent. In both cases the convention should be adopted that:

- Alternate diameter (ADIAM) is only entered in years when measurements are made at both the old and new POMs. In years where only one measurement point is used, it is entered in the DIAM field (column B on the spreadsheet).

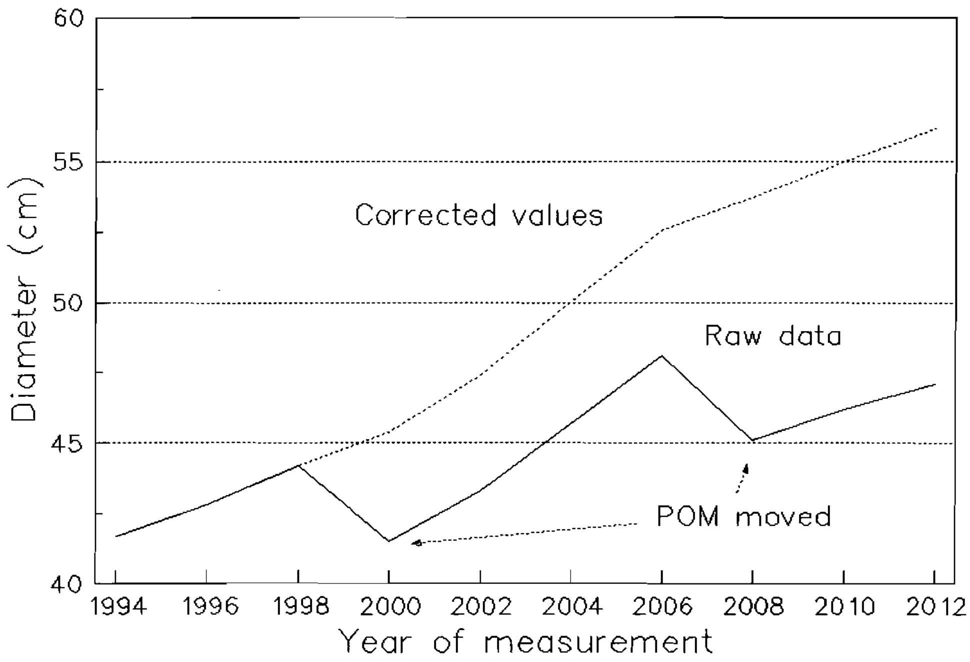


Figure 7 ***Raw diameter measurements and corrected values with moving POM***

- The POM field is likewise only entered in years when the ADIAM field is used. Otherwise it is assumed to be the same as for the preceding year. This assumption is not critical to the calculation of corrected diameters by the ratio method to be described, but if adhered to, assists the extraction of data for buttress taper equation development.

- In the first year of data collection, the initial point of measurement is recorded. This will commonly be at the standard height of 1.3 m ('breast height'), but its entry in the database will help to avoid ambiguity.

The simplest method for correcting for the movement of the point of measurement is the ratio method. In this, measurements subsequent to a move are multiplied by the ratio of the original diameter over the alternate diameter. This is shown in the spreadsheet by the formula for column E, the correction ratio. This states that:

$$\text{If } C_{r-1} > 0 \text{ then } E_r = E_{r-1}.B_{r-1}/C_r \text{ else } E_r = E_{r-1}$$

where C is the column containing alternate diameters, E is the correction ratio column, and B is the diameter at the previous or standard POM. The current row is r and the previous row is r-1. The standard reference diameter, or corrected diameter, is then calculated as the measured diameter in the B column or DIAM field times the correction ratio.

Figure 7 compares the original diameter measurements as they would be if no allowance where made for a changing POM and the calculated corrected diameters.

```
* inserts diameters corrected for a changed POM in field DREF
parameters pspdata
set talk off
use (pspdata) order tag pqty in 1
treeid0='?'
* examine data records in sequence
scan
  * ID of this tree
  treeid1=str(PLOTID,4)+str(QNO,2)+str(TNO,4)
  * see if same as preceding line of data
  if treeid1=treeid0
    * see if POM has moved
    if ADIAM>0
      * adjustment to ratio to be used with next record
      radj=DIAM/ADIAM
    else
      radj=1.0
    endif
  else
    * reset ratio for a new tree
    radj=1.0
    ratio=1.0
  endif
  * update treeid, fill DREF field
  treeid0=treeid1
  replace DREF with DIAM*ratio
  ratio=ratio*radj
endscan
set talk on
return
```

Example 14 *Program to calculate a standard reference diameter with moving POM*

An XBASE program MOVEPOM, to carry out these calculations on a database is shown in Example 14. The logic which detects the start of a new tree in the database is similar to that of the error checking program in Example 8. The program requires as a preliminary that an additional field is added to the PSPCD database called DREF, which contains the corrected reference diameter. This can be done via the XBASE MODIFY STRUCTURE command.

Another common approach that has been adopted to this problem in some increment studies is not to correct the diameter for changing POM at all, but simply to calculate increments based on diameters at different heights. In this case, using the data entry conventions noted above, increment is best calculated as the difference between the DIAM fields in successive measurements when the preceding ADIAM field is empty, but as ADIAM-DIAM when the preceding ADIAM field contains a value. This will give a series of increasing increments (barring instrument error or tree shrinkage), but they will relate to different heights up the tree. The increments obtained will be lower than those derived using a corrected diameter at standard reference height.

2.5.3 *Conversion of diameter series to increment files*

Much of the analysis on PSP data sets is concerned with deriving functions or transition matrices based on tree increments, expressed in terms of diameter or sectional area at a standard reference height. The file formats so far discussed are inconvenient for regression analysis as it is difficult for standard statistical packages to process them in the way required. The data needs to be manipulated so that each record contains:

- The essential basic predictor variables required for analysis. This may

```
procedure incxt
* extracts increment and other data for regression analysis from psp data
parameters ipf,opf
set talk off
set status off
select 1
use (ipf) order tag pqty alias ipf in 1
use (opf) alias opf in 2
treeid0='?'
* examine data records in sequence
scan
  * ID of this tree
  treeid1=str(PLOTID,4)+str(QNO,2)+str(TNO,4)
  * see if same as preceding line of data
  if treeid1=treeid0
    * calculate increment in cm/yr
    nyr=(AYR-year0)/365.25
    dinc1=(DREF-diam0)/nyr
    * generate output file record
    select opf
    append blank
    replace PLOTID with ipf->PLOTID, TNO with ipf->TNO, SPP with ipf->SPP, ;
            YEAR with year(ipf->AYR), DREF with ipf->DREF, DINC with dinc1, ;
            CFS with ipf->CFS, TCI with compix()
    select ipf
  endif
  * update treeid, species, diam for a new tree
  treeid0=treeid1
  diam0=DREF
  year0=AYR
endscan
set talk on
set status on
return

function compix
* calculates a tree competition index in variable tcix
  ...
  ...
return tcix
```

Example 15 *Program to extract diameter increment and other statistics* include tree diameter, crown class, species code, tree and plot number for reference.

- Derived indices of competition. These may be aggregate stand statistics such as quadrat basal area, or spatial measures using a number of possible methods.

- The tree increment, expressed over a standard time unit (eg, 1 year).

The program shown in Example 15 extracts these data from a PSP data file to a new file which should be created to include the following fields:

PLOTID	Plot identification number
TNO	Tree number
SPP	Species code
YEAR	Year measured
DREF	Corrected reference diameter
DINC	Diameter increment in cm/yr
CFS	Crown Freedom Score
TCI	Tree competition index

The PSP data file to be operated on by the program should have a structure similar to that shown on page 25 but with inclusion of the corrected DREF field as discussed in section 2.5.2.

The output file is intended purely for analysis of increment functions, and not for mortality or ingrowth studies. Specialised files and programs for these purposes are considered in sections 4.3 and 4.4.

The example program is similar to that shown in Example 8 (page 22) which also calculates tree increments for the purpose of range checking and error detection. In the present case however, the calculated values are saved to the output file using the APPEND BLANK... REPLACE... sequence. The calculation of tree competition index TCI is delegated to a user defined function COMPIX and is not explicity shown here. Several possible methods are given in section 2.6, where the subject of tree competition is discussed in detail.

This program requires that both the input and output files be specified as parameters when it is called from XBASE. It could be executed for example with the command:

```
DO INCXT WITH "PSPCD","DIAMINCS"
```

where DIAMINCS would contain the generated output records.

2.5.4 *Changes to the tree identification format*

It is not unusual, especially with older PSPs, for the method of plot, tree, or quadrat numbering to be awkard or inconveniently complex. One program is demonstrated here to show the type of techniques that may be applied to make modifications.

The Kade PSP data, part of which is shown listed in Figure 4 had originally a combined plot and tree number tagged for each tree, such as A101, meaning tree number 101 in plot A. This was further complicated by the use of a variety of decimals and year prefixes to indicate ingrowth trees. The system did not allow simple indexing by plot and tree identification in XBASE.

```
 1   * fills in the PLOTID and TREENO fields in file KADE
 2   set talk off
 3   use kade
 4   letters="ABCDEFGH"
 5   n=1
 6   do while n< =len(letters)
 7     L=substr(letters,n,1)
 8     replace plotid with L for L $ treeid
 9     n=n+1
10   enddo
11   index on plotid tag plotid
12   n=1
13   do while n< =len(letters)
14     L=substr(letters,n,1)
15     seek L
16     t=1
17     scan while plotid=L
18       replace treeno with t
19       t=t+1
20     endscan
21     n=n+1
22   enddo
23   set talk on
```

Example 16 *Program ADDID to change tree and plot numbering system in the Kade PSP data*

To rectify it, a small program called ADDID was written which is shown in Example 16. Before running the program, the original database was modified to include a field PLOTID, to contain a single letter plot identification (A-H), and one called TREENO, to contain a new sequential tree number.

The loop in lines 6-10 of the program fills in the PLOTID field for each plot in turn, from A to H. The plot letter is selected with the substring function at line 7, and then all trees whose original ID contains that letter have the PLOTID replaced with the letter. The XBASE operation 'L $ TREEID' gives a logical value True if the characters in variable L occur anywhere within the string in the field TREEID.

The trees are then indexed on the newly assigned PLOTID in line 11.

The outer loop in lines 13-22 scans through each plot letter in turn, from A to H, by selecting them from the string LETTERS using the pointer N, which will take values from 1 to 8 as it is incremented.

For each plot, the SEEK operation at line 15 positions the record pointer at the first record of the plot. The SCAN loop in lines 17-20 replaces each value of TREENO with the value of T. This is incremented for each tree, giving a sequential series of numbers within the plot.

It is then possible to index this database on plot and tree number in order to undertake the various other manipulations described in this chapter.

2.5.5 *Creating a database of plot-level statistics*

For a variety of purposes it is useful to calculate plot-level statistics such as basal area, numbers of trees, plot volume, and other possible measures. Such data may be used, for example, to:

- Derive competition indices based on plot basal area
- Give a means of site classification
- Provide data for whole-stand models.

One way to calculate basal area and stem numbers for each plot is to use the XBASE TOTAL command*. To do this, additional fields should be added to the PSPCD database, and an alternative index created that discriminates plots and measurement years, but not trees.

The following stages would be involved:

(i) Start with a series of sequential tree measurements such as the PSPCD database shown on page 25. Make a back-up copy of the database before commencing the modifications described below.

(ii) Add the corrected diameter field DREF as described in section 2.5.2 and calculate it using the MOVEPOM program in Example 14.

(iii) Using the MODIFY STRUCTURE command, add three additional fields called BA, TALLY, and PLOTY. The revised structure of the PSPCD file will then appear as shown in Example 17. The index flag for the PLOTY field should be set to Y in the interactive MODIFY STRUCTURE command.

(iv) Fill in the BA, TALLY and PLOTY fields. The BA field is given the tree basal area, calculated from DREF, and the TALLY field is set to 1. The PLOTY field controls the subsequent TOTAL command and should contain the plot number concatenated with the year of measurement, both in string format. The following command will achieve this:

* The SQL SELECT command provides an alternative method available with some XBASE dialects. Examples using this command to summarise data are given elsewhere in this manual.

Field	Type	Length	Usage
PLOTID	N	4	Plot identification number
AYR	D	8	Date of assessment
QNO	N	2	Quadrat number
TNO	N	4	Tree number
SPP	N	4	Species code
LOCE	N	4.1	Tree location East
LOCN	N	4.1	Tree location North
DIAM	N	5.1	Diameter measurement
ADIAM	N	5.1	Alternate diameter at new POM
POM	N	5.2	Point of measurement
CFS	N	1	Crown Freedom Score
NOTES	C	20	Codes notes, etc.
CHKSUM	N	6.1	Checksum
DREF	N	5.1	Corrected reference diameter (calculated)
BA	N	8.4	Basal area (calculated)
TALLY	N	8	Used for tree tallies with TOTAL command

Example 17 *Extended structure of PSPCD database with fields for basal area and tree count calculations*

```
REPLACE ALL BA WITH DREF*DREF*0.00007854, TALLY WITH 1, ;
        PLOTY WITH STR(PLOTID,4)+STR(YEAR(AYR),4)
```

(vi) Set the PLOTY index active and total the BA and TALLY fields within plots for each measurement year with the commands:

```
SET ORDER TO TAG PLOTY
TOTAL ON PLOTY TO PLOTSUM FIELDS BA,TALLY
```

The database PLOTSUM will have been created with plot basal areas in the BA field, and total tree counts in the tally field. Note that the structure of the PLOTSUM database will be identical to that of the input PSPCD file. However, the contents of the fields other than PLOTID, AYR, BA, TALLY and PLOTY will refer only to the first tree on each plot and should be discarded. The PLOTY field itself replicates the information in PLOTID and AYR, although the full assessment date, given in AYR, is abbreviated to give the year only in PLOTY. The surplus information can be discarded by using a selective copy command such as:

```
USE PLOTSUM
COPY TO PSPBA FIELD PLOTID,AYR,BA,TALLY
```

The resulting file PSPBA will contain only the following fields:

PLOTID	Plot identification number
AYR	Assessment date
BA	Plot basal area
TALLY	Count of trees on the plot

These methods of deriving plot statistics use only interactive commands and can therefore be executed from the menu-driven interfaces provided with various

XBASE implementations. They will be found to be slow with large files; much more efficient procedures can be devised with some programming. Some relevant concepts are discussed in section 2.5.7.

The SQL SELECT command, on those systems where it is available, is simpler and more efficient. An example is given in the competition influence overlap zone program CIOZ discussed on page 59. It allows complex calculations and summaries on grouped data in a single command, and in the better SQL implementations, is highly optimized for execution speed.

2.5.6 *Tallying methods for extracting preliminary data statistics*

It may be useful to monitor the species and size distribution of sampling during a programme of PSP work. The TALLY field described in the preceding section can be used with the TOTAL command to do this.

To monitor species distribution, proceed as follows:

(i) Using the PSPCD file as modified in the preceding section (see Example 17), copy the SPP and TALLY fields to a temporary database for a particular assessement year and set of plots of interest. Suitable commands would be:

```
USE PSPCD
COPY TO SPPTALLY FIELDS SPP,TALLY FOR YEAR(AYR)=1994
```

(ii) Open the resultant file of species and tallies, index on the species code, and total by species to a file called SPPSUM:

```
USE SPPTALLY
INDEX ON SPP TAG SPP
TOTAL ON SPP TO SPPSUM
```

The SPPSUM file will contain species codes and counts of trees. This can be viewed as a histogram by importing it into a package such as SYSTAT or Lotus 1-2-3. Some database packages, such as Paradox, will allow the histogram to be created directly from the data.

Field	Type	Length	Usage
SPP	N	4	Species code
DREF	N	5.1	Diameter corrected to reference height
DCLASS	N	2	Diameter class number
SPPD	C	7	Concatenated value of SPP and DCLASS used to control TOTAL
TALLY	N	8	Used to count trees. initially set to 1 for each record.

Example 18 *Structure of intermediate file to summarize data by species and diameter class*

It is also useful to consider the diameter distribution of the data, perhaps in combination with species as a 2-way table. This can be done by, for example, creating an intermediate database called SPPDC with the structure shown in Example 18. Selected fields from the PSPCD database are then copied for a particular measurement year, using commands similar to the following:

```
USE SPPDC
APPEND FROM PSPCD FOR YEAR(AYR)=1994
```

The DCLASS and SPPD fields are filled in with commands like:

```
REPLACE DCLASS WITH INT((DREF-10)/20)+1, ;
  SPPD WITH STR(SPP,4) + STR(DCLASS,3)
```

The method of deriving a diameter class from a diameter value is quite general and is used widely in different programs in this book. If the class width is w, and the lower bound of the first class is f, then the class index k for a given diameter d will be given by:

$$k = \text{int}\{(d-f)/w\}+1 \qquad \{1\}$$

where the **int()** function truncates a number to the next lower integer. Thus for example, if the class width is 20 cm and the lowest class starts at 10 cm, a tree of diameter 34.3 would be:

$$k = \text{int}\{(34.3-10)/20\}+1$$

which evaluates to 2.

The file is then indexed on the SPPD field and totals calculated:

```
INDEX ON SPPD TAG SPPD
TOTAL ON SPPD TO SPPDCSUM
```

In the database SPPDCSUM, the TALLY field will then contain the tree count for the selected data for each species and diameter class.

The manipulations described in this and the preceding section are designed to allow simple interactive methods of reviewing some aspects of the data. However, they will be found to be slow for large data sets comprising several tens or hundreds of PSPs. For the latter case, programmed methods of analysis using hash tables and arrays are faster than using INDEX, REPLACE and TOTAL commands. These latter methods are illustrated in the following sections.

2.5.7 *Reducing large files by cross-tabulation*

In a typical case with perhaps 100 PSPs, each containing about 500 trees and 3 re-measurements, there would be a total of 100,000 tree increment observations. This is too large a body of data to be conveniently analysed by most regression packages. It is therefore necessary to condense the data in some way without losing essential information.

```
Structure for database: D:\GHPSPINC.DBF
Number of data records:    12788
Date of last update   : 08/07/93
Field  Field Name  Type        Width    Dec    Index  Description
    1  FRESV       Character     4                 N  Forest reserve code
    2  PNUM        Character     3                 N  Plot number
    3  LDNO        Character     3                 N  Leading Desirable (tree) number
    4  YR1         Numeric       2                 N  First year measured
    5  YR2         Numeric       2                 N  Last year measured
    6  SPP         Character     3                 N  Species code
    7  DPOM        Numeric      10        2        N  Diameter at POM, cm
    8  DINC        Numeric      10        2        N  Increment cm/yr
    9  CPOS        Numeric       5                 N  Crown score
   10  NBA         Numeric       8                 N  No. of competing trees
   11  DEAD        Character     2                 N  Dead or missing tree codes
   12  TALLY       Numeric       5                 N  Used for preliminary statistics
** Total **                     58


FRESV PNUM LDNO YR1 YR2 SPP         DPOM        DINC  CPOS        NBA DEAD TALLY
ASUK     5    2  72  82 Eu         10.91        0.10     2          5          1
ASUK     5    3  72  82 Ant        90.35        1.24     5          4          1
ASUK     5    5  72  82 Tri        31.73        1.06     4          4          1
ASUK     5    6  72  82 Ec        128.75        0.00     0          0 NT       1
ASUK     5    7  72  82 Man        45.68        0.45     4          5          1
ASUK     5    8  72  82 Eu          7.98        0.00     0          0 D        1
ASUK     5    9  72  82 Ge         22.23        0.00     0          0 D        1
ASUK     5   10  72  82 Ge         14.25        0.03     2          4          1
ASUK     5   11  72  82 Ec          4.24        0.20     2          3          1
ASUK     5   12  72  82 Tri        23.55        2.05     4          2          1
ASUK     5   13  72  82 Ant        11.32        0.00     0          0 D        1
ASUK     5   14  72  82 Eu         13.04        0.00     0          0 D        1
ASUK     5   15  72  82 Eu         29.71        0.12     3          2          1
ASUK     5   16  72  82 Nes        13.74        0.32     2          3          1
ASUK     5   17  72  82 Tri        60.03        0.37     4          5          1
```

Example 19 *PSP tree increment data to be processed by cross-tabulation program*

The most suitable way to achieve such a summary is to cross-tabulate the data by the parameters to be used as predictors in a regression equation. The procedure is described here for a simple and typical case that would be involved in developing a basic stand projection model. The data is to be classified to give mean increments and mortality rates by diameter classes for each species sampled.

Example 19 shows the structure of the input file for cross-tabulation. It relates to PSPs established in Ghana between 1968 and 1983 on which selected 'Leading Desirable' trees were measured (Baidoe, 1968). The program listed in Example 20 performs the cross tabulation. The program operates as follows:

- Line 3 suppresses the normal XBASE interactive responses while the program is running.

- Lines 5 and 6 open the input and output files. The input file must be indexed by the species code, SPP.

```
* cross-tabulation program to summarise PSP statistics
* by diameter classes and species
set talk off
* open psp file (p) and summary file (q)
use ghpspinc order tag SPP alias p in 1
use incsum alias q in 2
* parameters of diameter classification. cw=width, fc=1st class, nc=no classes
cw=10
fc=0
nc=11
declare incsum[nc],incsumsq[nc],ntrees[nc],ndead[nc],dsum[nc]
select p
* outer loop scans by groups of species
scan
  * reset species code, accumulators to zero for next species
  spp0=SPP
  k=1
  do while k<=nc
    store 0 to incsum[k],incsumsq[k],ntrees[k],ndead[k],dsum[k]
    k=k+1
  enddo
  * scan through species
  scan while SPP=spp0
    * get diameter class, count trees, sum diameters
    k=int((DPOM-fc)/cw)+1
    k=iif(k>nc,nc,k)
    ntrees[k]=ntrees[k]+1
    dsum[k]=dsum[k]+DPOM
    * add increments and sums of squares for live trees, count dead ones
    if DEAD=' '
      incsum[k]=incsum[k]+DINC
      incsumsq[k]=incsumsq[k]+DINC*DINC
    else
      ndead[k]=ndead[k]+1
    endif
  endscan
  * output results, converting totals to means, sumsq to standard error
  k=1
  select q
  do while k<=nc
    n=ntrees[k]
    if n>0
      append blank
      se=iif(n>1, sqrt((incsumsq[k]-incsum[k]^2/n)/(n-1)/n), 0)
      replace SPP with spp0, DCLASS with k, DMEAN with dsum[k]/n, DINC with ;
              incsum[k]/n, INCSE with se, TREES with n, DEAD with ndead[k]
    endif
    k=k+1
  enddo
  select p
endscan
set talk on
return
```

Example 20 *Listing of XTAB program to summarise increment data by species and diameter class*

- Lines 8-10 define the diameter classes. CW is the class width in cm, FC the lower bound of the first class, and NC is the number of classes.

- Line 11 declares 5 arrays used to accumulate data by diameter classes within a species. INCSUM contains totals of increments, INCSUMSQ contains sums of squares, NTREES counts the total trees sampled (both dead and alive), NDEAD counts the dead trees, and DSUM totals the tree diameters.

- Line 14 begins a loop which at ends at line 51 to scan through blocks of data by species.

- Lines 16-21 set the species code at the first line for a block of data for a new species to SPPO. The end of the species block is detected when SPPO no longer equals the species code on the current record, as given by the WHILE clause in line 23. The arrays are all reset to zeroes to initialize them for the accumulation of within-species totals.

- Line 23 starts the within-species scan through the data which ends at line 36. This loop totals the various statistics required.

- Lines 25-26 calculate a class number from the tree diameter. Trees over 100 cm are all placed in class 11.

- Lines 27 to 35 accumulate the various statistics required.

- Lines 38-49 output the species totals to the database file INCSUM, aliased as Q. One record is created in the output file for each diameter class. The sum of squares of increments are converted to standard error at line 44, whilst other totals are converted to means in the REPLACE statement itself at lines 45-46.

The structure for the output file is shown in Example 21. This structure must be created in XBASE before the XTAB program is run. If XTAB is to be run several times, then the INCSUM datbase must be emptied before running the program by using the ZAP command with the INCSUM file in use in the currently selected work area.

It will be noted in this example that mnemonic species codes were used, rather than numeric ones. The program will function identically with numeric codes.

The arrangement of the data with one line per diameter class is generally more flexible than one in which all diameter classes are arranged in tabular format across a record. The latter arrangement, although more suitable for a printed presentation of the results, is difficult for regression and graphical packages to process. The layout shown can also be extended to multi-way classifications involving, for example, species, diameter class, and competition class.

The effectiveness of the cross-tabulation process in reducing the size of the data

```
Structure for database: D:\INCSUM.DBF
Number of data records:     202
Date of last update   : 20/08/93
Field  Field Name  Type        Width    Dec    Index
    1  SPP         Character       3                Y  Species code
    2  DCLASS      Numeric         2                Y  Diameter class no.
    3  DMEAN       Numeric         5      1         N  Mean diameter in class
    4  DINC        Numeric         5      2         N  Mean increment of class
    5  INCSE       Numeric         5      3         N  Standard error of increment
    6  TREES       Numeric         5                N  Total trees sampled
    7  DEAD        Numeric         4                N  No. of trees which died
** Total **                       30


SPP DCLASS DMEAN  DINC INCSE TREES DEAD
Ea       1   4.6  0.22 0.046   219   24
Ea       2  14.5  0.22 0.013   258    8
Ea       3  24.1  0.28 0.061   174   10
Ea       4  34.7  0.39 0.041    71    1
Ea       5  44.7  0.40 0.061    21    1
Ea       6  56.0  0.50 0.155    12    3
Ea       7  64.5  0.62 0.110    12    0
Ea       8  73.0  0.54 0.180     3    0
Ea       9  84.7  0.30 0.095    10    3
Ea      10 100.0  0.00 0.000     1    1
Ea      11 118.8  0.15 0.147     3    2
Ec       1   3.9  0.33 0.081   216   22
Ec       2  15.1  0.30 0.046   224    8
Ec       3  24.4  0.39 0.030   140    6
Ec       4  34.1  0.71 0.137    45    2
Ec       5  44.7  0.69 0.067    38    1
Ec       6  55.1  0.64 0.082    20    0
Ec       7  65.5  0.86 0.121    22    2
Ec       8  74.4  0.40 0.109    14    4
Ec       9  84.3  0.56 0.112    10    2
```

Example 21 *Structure and partial listing of summary database produced by cross-tabulation program XTAB*

set to be manipulated can be seen. The raw data file shown in Example 19 has 12,788 records, whereas the summary file in Example 21 has only 202 records.

2.5.8 *Converting multi-year record formats to year-per-record files*

Section 2.3.5, page 24, has discussed the nature of multi-year record (MYR) format files, as opposed to year-per-record (YPR) formats. For regression analysis and for most of the other manipulations described in this manual, it is desirable to convert MYR to YPR format. This section illustrates the general approach, using as an example the database for the Kade PSPs, part of which is shown in Figure 4. The program illustrated in Example 22 shows a typical XBASE program for this type of reformatting. Lines 1-6 of the program organize preliminaries, clearing the status line and screen, closing any open databases, giving a program title at the top of the screen, and setting off the interactive responses to commands.

Lines 7 and 8 open the input and output databases and give them the aliases MYR and YPR respectively. The alias is a short name used in the program to reference the file work area.

Line 9 creates an array whose 14 elements will correspond to the 14 fields in the input MYR records. The MYR database is made active and a loop, encompassing lines 11-40, begins which scans each record of the input database in turn and processes it.

```
 1  set talk off
 2  set status off
 3  clear
 4  @ 0,0 say "MAKEYPR : Converts Kade PSP data from MYR to YPR format"
 5  @ 1,0 to 1,70 double
 6  close databases
 7  use kade alias myr in 1
 8  use kadeypr alias ypr in 2
 9  declare amyr[14]
10  select myr
11  scan
12    copy to array amyr record recno()
13    @ 3,0 say "MYR input record no.  "+str(recno(),5)
14    yf=7
15    select ypr
16    do while yf< =14
17      do case
18      case amyr[yf] = -1
19        * small size trees < 300 mm girth - nothing written to output file
20      case amyr[yf] > 0
21        * normal girth measurement, copy to output file as dbh
22        append blank
23        replace PLOTID with myr->PLOTID, TREENO with myr->TREENO, ;
24                SPPNO with myr->SPPNO, XCOORD with myr->XCOORD,   ;
25                YCOORD with myr->YCOORD, MYEAR with val(right(field(yf,[myr]),2)), ;
26                DBH with amyr[yf]/31.4159, DEAD with .F.
27      case amyr[yf] = -99
28        * tree has died.  use dbh from previous year and set DEAD true
29        append blank
30        replace PLOTID with myr->PLOTID, TREENO with myr->TREENO, ;
31                SPPNO with myr->SPPNO, XCOORD with myr->XCOORD,   ;
32                YCOORD with myr->YCOORD, MYEAR with val(right(field(yf,[myr]),2)), ;
33                DBH with amyr[yf-1]/31.4159, DEAD with .T.
34        exit
35      endcase
36      @ 4,0 say "YPR output record no. "+str(recno(),5)
37      yf=yf+1
38    enddo
39    select myr
40  endscan
41  set talk on
42  set status on
43  return
```

Example 22 *Program to re-organize data from multi-year records to year-per-record (YPR) format*

At line 12, the fields of the current record of the MYR database are copied to the array AMYR. Line 13 gives the record number on screen to appraise the user of the program's progress. Line 14 is a pointer to the first time-dependent field, GIRTH68, which is stored in the 7th field of the MYR database.

The output database is made the active file at line 15, and loop begins through the time-dependent fields GIRTH68 to GIRTH93, which are stored in the array AMYR, elements 7 to 14. Inside the loop, three possible cases for the value of girth are

```
Original MYR format

 PLOTID TREENO TREEID   SPPNO XCOORD YCOORD GIRTH68 GIRTH70 GIRTH72
 A         149 A7.10     57     97     67       -1      -1      -1
 A         150 82A1      94     68     91       -1      -1      -1
 A         151 82A2      65     61     79       -1      -1      -1
 A         152 82A3      22     79     79       -1      -1      -1
 A         153 82A4      16     75     79       -1      -1      -1

                               GIRTH77 GIRTH80 GIRTH82 GIRTH87 GIRTH93
                                   310     311     310     320     320
                                    -1      -1     330     345     361
                                    -1      -1     320     345     -99
                                    -1      -1     355     415     510
                                    -1      -1     310     325     360


Converted YPR format

 PLOTID TREENO SPPNO XCOORD YCOORD MYEAR  DBH DEAD
 A         149   57     97     67    77   9.9 .F.
 A         149   57     97     67    80   9.9 .F.
 A         149   57     97     67    82   9.9 .F.
 A         149   57     97     67    87  10.2 .F.
 A         149   57     97     67    93  10.2 .F.

 A         150   94     68     91    82  10.5 .F.
 A         150   94     68     91    87  11.0 .F.
 A         150   94     68     91    93  11.5 .F.

 A         151   65     61     79    82  10.2 .F.
 A         151   65     61     79    87  11.0 .F.
 A         151   65     61     79    93  11.0 .T.

 A         152   22     79     79    82  11.3 .F.
 A         152   22     79     79    87  13.2 .F.
 A         152   22     79     79    93  16.2 .F.
```

Example 23 *Original MYR and converted YPR formats for several trees from the Kade data*

examined, and appropriate action taken.

Trees which were too small to measure (<300 mm gbh) are coded with a girth of -1. This cases is detected at line 19 and no output is written.

The normal case of a positive girth measurement is detected at line 20. A blank output record is created, and the fields filled in with the REPLACE command at line 23. The time-invariant fields PLOTID to YCOORD are copied directly from the input record. The year is calculated from the two right-most characters of the GIRTH field name currently being operated on and stored in MYEAR. DBH in cm is derived from girth in mm by dividing by 10π. The DEAD field is set as false.

Trees which have died are detected at line 27; they have the girth field set to -99. They are coded similarly to the live trees except that the DBH is copied from the preceding field to the left, and the DEAD field is set to true. The program then exits from the loop, skipping any girth fields to the left after the tree has died.

At line 36, status information on the output file is displayed for the user. Line 37 increments the current girth field. The loop terminates when YF is 15, that is after operating on the field GIRTH93.

At line 39, the input file is reselected. The outer loop is repeated from line 40, incrementing the record number until the end-of-file is encountered. Lines 41 and 42 reset normal interactive operation, and the program returns control to the user at line 43.

The input and output file formats are shown in Example 23 for several trees.

2.5.9 *Conversion of database files to generic formats*

For many purposes, it is possible to work directly with database files, either from within the database package, or using spreadsheet, graphical or statistical programs that can read the database format being used. However, in some cases, it may be necessary to convert the data into a generic ASCII format so that it can be read by programs written in general purpose programming languages such as C or BASIC.

There are two common generic formats for database files:

- Fixed-width fields without separators. In XBASE, these are known as SDF (System Data Format) files.
- Comma-separated value (CSV) files, in which each field is separated by a comma, and text fields are enlosed in quotation marks.

```
"Ant",1,4.7,0.30,0.114,122,35
"Ant",2,14.8,0.29,0.032,173,14
"Ant",3,24.8,0.26,0.041,119,11
"Ant",4,35.0,0.46,0.060,60,3
"Ant",5,43.2,0.55,0.089,28,1
"Ant",6,53.7,0.61,0.102,23,1
"Ant",7,64.0,0.94,0.131,21,1
"Ant",8,74.8,0.59,0.116,14,1
"Ant",9,83.7,0.61,0.109,13,2
"Ant",10,95.3,0.98,0.239,14,1
"Ant",11,105.5,0.41,0.095,14,2
"Chl",1,7.3,0.31,0.101,8,2
"Chl",2,16.3,0.36,0.094,25,1
"Chl",3,25.4,0.36,0.048,38,2
"Chl",4,33.9,0.40,0.106,18,1
"Chl",5,44.1,0.46,0.092,16,0
"Chl",6,54.0,0.21,0.052,12,0
"Chl",7,65.4,0.04,0.035,2,1
"Chl",8,73.4,1.17,0.491,4,1
"Chl",9,84.6,0.21,0.071,7,1
"Chl",10,94.1,0.32,0.323,3,2
"Chl",11,109.2,0.12,0.096,6,3
```

Example 24 *Part of the INCSUM database listed in CSV format*

Both types of file are easily created from XBASE using the COPY TO command. An indexing method should be selected that will produce the records in the most convenient sorted order for analysis, as sorting operations within a statistical package or user program to re-order the records may be slow. Example 24 shows how the INCSUM summary database discusssed in the preceding section appears when listed in CSV format. This file is created with the following XBASE commands:

```
USE INCSUM
COPY TO INCSUM.CSV TYPE DELIMITED
```

For programs written in BASIC, the CSV format is undoubtedly the simplest to use, as it can be read using the standard INPUT statement. For programs in FORTRAN, SDF files can be processed with formatted READ statements.

2.6 Calculation of competition indices

2.6.1 *Methods based on quadrat and plot statistics*

Competition indices based on plot or quadrat parameters do not require a knowledge of the positional relationships of trees, and are therefore simpler to use than those that require tree coordinates.

Vanclay (1994) discusses some alternative indices that have been used. From the point of view of data analysis, these can be grouped as follows:

(i) **Stand basal area**, calculated at the plot or quadrat level. For the purposes of evaluating the effectiveness of basal area as a competition index, it may be desirable to calculate both.

(ii) **Relative tree size**. This may include tree diameter or sectional area relative to the largest tree or to the mean basal area tree on a plot or quadrat.

(iii) **Overtopping Basal Area** (OBA). This is defined as the basal area of all trees on the plot larger than the subject tree. OBA can be calculated at the quadrat or plot level, and it may often be useful to compare both indices.

At the early stages of analysing PSP data it will not be known which competition indices are likely to prove most useful. It is therefore appropriate to create a file that includes tree size, increment, observed competitive status (crown freedom score, etc.) and several calculated competition indices. These may then be compared with increment by regression analysis to determine which index is most useful.

Later, the programming and analysis may be simplified by using only a single preferred competition index.

To illustrate the procedures involved, the year-per-record file KADEYPR shown in Example 23 is used. The first stage is to calculate plot-level statistics. An output file for the plot data is created with fields for plot ID, measurement year (MYEAR), basal area (SBA), number of trees (NTREES), and diameter of largest tree (DMAX). Diameter of the tree of mean basal area (DG) is also included as a derived field. The database structure is shown in Example 25.

Although the TOTAL command can be used directly in XBASE to derive tree numbers and basal area, as described in section 2.5.5, it is not possible to calculate DMAX in this way. An alternative approach is shown in the program in Example 26. This uses an index called YP to organize the data by years and plots. YP is created prior to program execution with statements such as:

Plot summary database KPSUM.DBF

Field	Field Name	Type	Width	Dec	Usage
1	PLOTID	Character	1		Plot ID, letter A-H
2	MYEAR	Numeric	2		Measurement year
3	SBA	Numeric	6	1	Basal area, m^2/ha
4	NTREE	Numeric	4		No. trees ≥300 mm gbh, N/ha
5	DMAX	Numeric	6	1	DBH of largest diameter tree, cm
6	DG	Numeric	6	1	DBH of tree of mean basal area, cm

Tree record database, extended to include competition indices and increment

Field	Field Name	Type	Width	Dec	Usage
1	PLOTID	Character	1		Plot ID, letter A-H
2	TREENO	Numeric	4		Tree number
3	SPPNO	Numeric	3		Species code number
4	XCOORD	Numeric	4		X coordinate, m
5	YCOORD	Numeric	4		Y coordinate, m
6	MYEAR	Numeric	2		Measurement year
7	DBH	Numeric	5	1	Tree DBH, cm
8	DEAD	Logical	1		True if tree has died since last measurement
9	TBA	Numeric	6		Tree basal area, cm^2
10	BAI	Numeric	5		Tree basal area increment, cm^2/yr
11	OBA	Numeric	5	1	Overtopping basal area, m^2/ha
12	SBA	Numeric	5	1	Stand basal area, m^2/ha
13	DRMEAN	Numeric	5	3	DBH/plot DG
14	DRMAX	Numeric	5	3	DBH/plot DMAX

Example 25 *Plot summary and tree increment database structures for position-independent competition index analysis*

```
USE KADEYPR
INDEX ON STR(MYEAR,2)+PLOTID TAG YP
```

The main features of this program are as follows:

Line Details

1 Standard interactive XBASE responses are switched off while the program is running. They are turned on again at the end, at line 33.

3-4 The databases to be used are opened in work areas 1 and 2. The controlling index of database KADEYPR will be YP, created as noted above.

5 The plot size is 0.25 ha. Dividing raw plot basal areas and stems numbers by PSIZE (lines 38, 42) gives statistics on a per ha basis.

7-9 The plot variables are initialized here. They are re-initialized after results for each plot are output at lines 21-23.

11-15 Sets up the screen display to track the progress of the program. Current values are written from lines 44-45 as each plot is output.

17-30 This scans the tree data record by record, accumulating plot data.

18-23 The end of a plot is detected when PLOTID for the current record is no longer the same as ID. At this point the procedure OUTRES is called to calculate and output plot results. The plot variables are then reset at line 21 and the new plot identification and measurement year made current.

```
 1  set talk off
 2  * open input tree and output plot files.
 3  use kadeypr order tag yp alias kt in 1
 4  use kpsum alias kp in 2
 5  psize=0.25
 6  * initialize plot variables
 7  store 0 to ba,nt,dbig
 8  id=PLOTID
 9  my=MYEAR
10  * screen display to show progress
11  @ 0,0 clear
12  @ 0,0 say "KPSUM : Plot summary program"
13  @ 1,0 to 1,79 double
14  @ 3,0 say "Year"
15  @ 4,0 say "Plot"
16  * scan tree data totalling to plots
17  scan
18    if PLOTID< >id
19      * start of new plot
20      do outres
21      store 0 to ba,nt,dbig
22      id=PLOTID
23      my=MYEAR
24    else
25      * accumulate within-plot data
26      ba=ba+DBH^2
27      nt=nt+1
28      dbig = iif(DBH>dbig,DBH,dbig)
29    endif
30  endscan
31  * output results for last plot
32  do outres
33  set talk on
34  return

35  procedure outres
36  * generate an output record in the summary file
37  xdg=sqrt(ba/nt)
38  ba=0.00007854*ba/psize
39  select kp
40  append blank
41  replace PLOTID with id, MYEAR with my, SBA with ba, ;
42          NTREE with nt/psize, DMAX with dbig, DG with xdg
43  select kt
44  @ 3,6 say my
45  @ 4,6 say id
46  return
```

Example 26 *Program KPSUM to summarize plot statisics including DMAX for competition index calculation*

26-28 Basal area is accumulated by summing DBH^2; trees are counted in NT. If the current tree is larger than DBIG, the present value if DBIG is replaced by the current value.

32 Outputs results for the last plot, when the end-of-file is encountered. This will have been missed by the test at line 18.

37 Calculates DG as $\sqrt{(\Sigma DBH^2/NT)}$. The variable name is prefixed with x to distinguish it from the field DG.

39-43 Makes the output file the active work area, appends a blank record to it, and fills in the fields with the plot values. The active work area is then switched back to the input file.

Once the plot statistics have been computed, then the tree competition indices can be derived in a fairly straightforward fashion. The original tree data file is first extended by adding new fields to take the competition values. The most efficient way to do this is to copy the structure of the original file to a new one, then modify it to add the new fields, and append the original data. This is safer than modifying the original file, and faster than copying the whole original file and then modifying it. The sequence of XBASE commands to create the modified file KTINC shown in Example 25 is as follows:

```
USE KADEYPR
COPY STRUCTURE TO KTINC
USE KTINC
MODIFY STRUCTURE
APPEND FROM KADEYPR
```

The final stage of the process is to run the XOBA program shown in Example 27. This comprises the following steps:

- Lines 1-5 and 61-65 are the standard opening and closing lines to set the environment for a XBASE program, and then reset it for interactive operation.

- The @ statements at lines 6-8, 19, 23, 41, 44, 52 and 60 monitor the progress of the program on screen. The K counter reset and incremented at lines 20, 22, 46, and 51 are used to display number of trees processed.

- The data files are opened and indexed at lines 10-15. The structure of these files is as shown in Example 25. The tree file is indexed on plot, tree number and measurement year in descending order. This will bring together all records for the same tree, starting with the most recent measurement and going back to the first observation. The file of plot level data, KPSUM, is ordered by plot and measurement year to allow look-up of values required for each tree.

- A loop through the records occurs at lines 21-39. This calculates increment and looks up the plot-level competition indices for each tree. Increment is not calculated for the first record of each series for a tree, which, because of the index order, will be the most recent observation.

  The IF statement at line 24, and the ELSE branch at lines 35-38 traps the situation where a tree record has no corresponding entry in the plot summary file. This will happen if the summary file has not been properly constructed.

- A new index is generated for the tree record file at line 42, resorting them by plot, year, and tree size, from largest to smallest. This is required for the OBA calculation.

```
set talk off
set status off
set scoreboard off
set safety off
clear
@ 0,0 say "XOBA : Calculates increment and competition indices"
@ 1,0 to 1,79 double
@ 3,0 say "Opening and indexing files ..."
* open tree record database and index for increment calculation
select 1
use ktinc alias kt
index on PLOTID+str(TREENO,4)+str(MYEAR,2) tag pty descending
select 2
use kpsum alias kp
index on PLOTID+str(MYEAR,2) tag py
select 1
treeid=[xxx]
* add tree BA, increment, plot BA and relative diameters
@ 4,0 say "Adding increments etc...."
k=0
scan
  k=k+1
  @ 4,30 say k
  if seek(PLOTID+str(MYEAR,2),[kp])
    tba1=DBH^2*0.7854
    if treeid=PLOTID+str(TREENO,4)
      replace TBA with tba1, BAI with tba2-tba1, SBA with kp->SBA, ;
              DRMEAN with DBH/kp->DG, DRMAX with DBH/kp->DMAX
    else
      replace TBA with tba1, BAI with -999, SBA with kp->SBA, ;
              DRMEAN with DBH/kp->DG, DRMAX with DBH/kp->DMAX
      treeid=PLOTID+str(TREENO,4)
    endif
    tba2=tba1
  else
    ? "Data error : Plot/year not in plot summary file"
    suspend
  endif
endscan
* change index to order from largest to smallest tree within year and plot
@ 5,0 say "Indexing for OBA ..."
index on PLOTID+str(MYEAR,2)+str(TBA,6) tag pyt descending
* add OBA for each tree
@ 5,0 say "Calculating OBAs ..."
goto top
k=0
id=PLOTID+str(MYEAR,2)
xoba=0
psize=0.25
scan
  k=k+1
  @ 5,30 say k
  if id< >PLOTID+str(MYEAR,2)
    xoba=0
    id=PLOTID+str(MYEAR,2)
  endif
  replace OBA with xoba
  xoba=xoba+TBA/10000/psize
endscan
@ 7,0 say "Finished OK"
close databases
set safety on
set talk on
set status on
return
```

Example 27 *Program XOBA to calculate increment, OBA and other competition indices*

- The loop at lines 50-59 calculates OBA for each tree. A flag, ID, changes with each new plot and measurement year. When it changes, the current OBA accumulator, XOBA, is reset to zero, and the ID reset for the next sequence (lines 53-56).

  The OBA value for the current tree is added to the file at line 57. It is then incremented to add the tree basal area, scaled to plot dimensions by PSIZE (plot area in ha, set at line 49). This sequence means that, because of the index order set at line 42, the largest tree will have an OBA of zero, the second largest will have an OBA corrsponding to one overtopping tree (the largest), the third will have two overtopping trees, and so on.

After completion of the XOBA program, the tree file will have fields for basal area, basal area increment, stand basal area, relative diameter, and OBA filled in and ready for statistical analysis. The further procedures for this are decribed in section IV.2.

2.6.2 *Competition indices based on tree position*

In principle, a model of tree growth which takes account of the crowding of a tree by its neighbours would appear likely to be highly effective. In practice, this has not generally been found to be the case, and the efficiency of such spatial competition indices may be no better than the much simpler indices discussed in the preceding section (Vanclay, 1994).

There are many possible ways a spatial competition index can be devised. Gourlet-Fleury (1992) provides a diagrammatic review of some of the methods used. Generally, methods may be 2-dimensional, 3-dimensional, or based on considerations of branch development and tree architecture. Only the 2-D methods will be discussed here. The other methods require height and crown dimension information which is only likely to be available for the most intensively measured experimental plots in TMF.

Generally, 2-D spatial competition indices can be divided into 3 categories as shown in Figure 8. These are as follows:

(i) *Growing space polygons (GSP)* (Adlard, 1974): For each tree, a line is connected to its neighbours, and a point determined on the line as a weighted function of the relative size of the trees. The points are joined to form a polygon around the subject tree whose area is a measure of the growing space available. In its simple form, this method will only work with uniform forests. With mixed forests, where trees are growing under the main canopy, and emergents may also be well above it, the polygons become very confused with re-entrant lines. A solution is to omit trees substantially smaller, by some pre-determined factor, than the subject tree.

(ii) *Competitive influence overlap (CIO) zones* (Bella, 1971; Ek & Monserud, 1974):

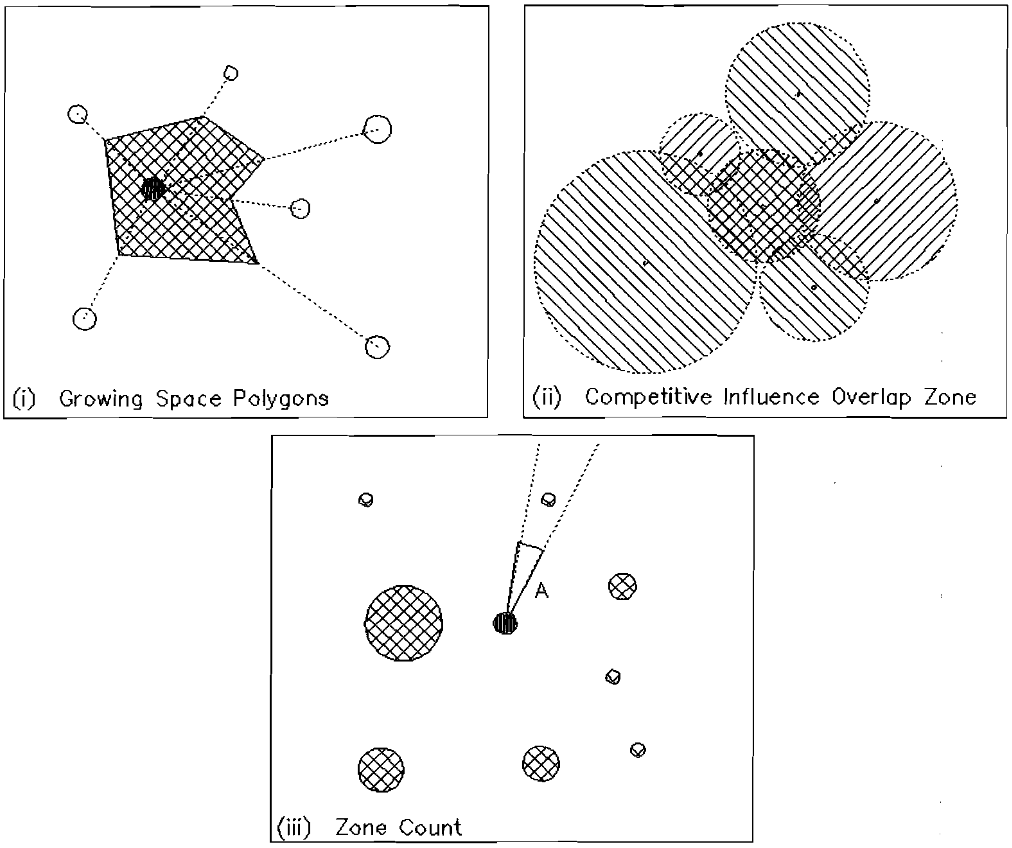

(i) Growing Space Polygons
(ii) Competitive Influence Overlap Zone
A
(iii) Zone Count

Figure 8 ***Basic methods for 2-dimensional competition indices based on tree position***

This is likely to be the most effective 2-D method, in terms of predictive power, and is also reasonably straightforward to calculate, not requiring the various geometric adjustments of the GSP method. Circles representing competitive influence are drawn around each tree. The radius of these circles may correspond to mean crown radius, or to some arbitrary size-weighted factor. For a particular subject tree, the total area of overlap, weighted by the relative size of the trees to represent height differences, provides a measure of competition. This is readily calculable in mixed forests, and understorey or emergent crowns do not pose any problem.

(iii) *Zone count (ZC)* (Opie, 1968): The zone count method uses some arbitrarily chosen angle to count the number of trees in a simulated basal area sweep around a subject tree. The method is simple in principle, but has not been much used in practice, probably because it is not very effective. To test the method, a number of different basal area factors (BAF), or sweep angles should be used to determine which gives the highest correlation with increment.

As the CIO methods are those most likely to be useful as competition indices, the calculations are considered in more detail, and an example program to calculate CIO indices presented. Given two adjacent trees, T_1 and T_2 with influence zones of radius r_1 and r_2 respectively, as shown in Figure 9, then the angles subtended from each tree to the mid-point of the common chord are given by:

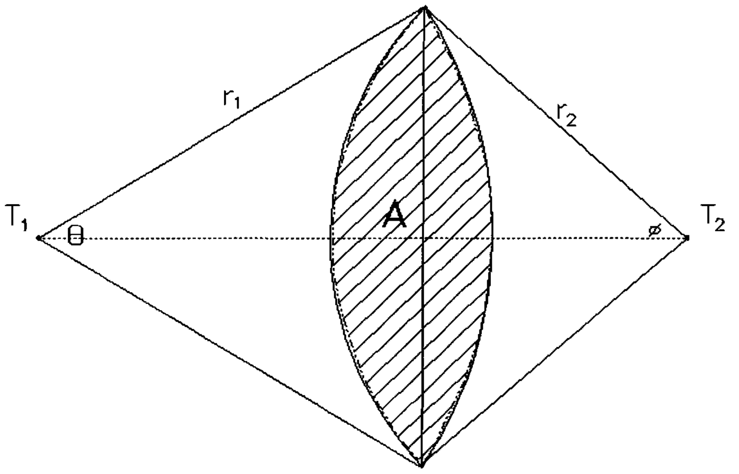


Figure 9 *Geometric basis for calculating overlap area of adjacent circles*

$$\Theta = \cos^{-1}((r_1^2 - r_2^2 + d^2)/(2dr_1)) \qquad \{2\}$$

$$\phi = \cos^{-1}((r_2^2 - r_1^2 + d^2)/(2dr_2)) \qquad \{3\}$$

The overlap area A is given by:

$$A = r_1^2.[\Theta - \sin(\Theta).\cos(\Theta)] + r_2^2.[\phi - \sin(\phi).\cos(\phi)] \qquad \{4\}$$

where:

r_1 is the radius of the influence zone of the subject tree;
r_2 is the radius of the influence zone of the competitor;
d is the distance between the two trees;
Θ is the angle *in radians* subtended from the subject tree to the mid-point of the common chord with the competitor;
ϕ is the angle *in radians* subtended from the competing tree to the mid-point of the common chord with the subject tree;
A is the area of the overlapping influence zones.

The above formulae are only applicable when $r_2 \leq d \leq r_1 + r_2$. The case where the influence zone of one tree is wholly inside another satisfies the condition that $d \leq |r_1 - r_2|$. In other words, the distance between the trees is less than the absolute difference between the radii of the influence zones. In such cases, the overlap area is given by the area of the influence zone of the smaller of the two trees, whether it is the competitor or the subject tree. This can be expressed algebraically as:

$$\text{If } d \leq |r_1 - r_2| \text{ then } A = \pi.min(r_1, r_2)^2 \qquad \{5\}$$

In the case where $d > r_1+r_2$, then the influence zones do not intersect, and the overlap area is zero.

The competition index for a given subject tree is given by:

$$CIO = (\Sigma A_k.w_k)/(A_s.w_s) \qquad \{6\}$$

where:

A_s is the area of the influence zone of the subject tree;
A_k is the area of the influence zone of the k'th competitor;
w_k is a weight for the k'th competitor, to take account of relative height with the subject tree, and also perhaps species factors such as foliage density or shade tolerance.
w_s is the weight for the subject tree.

In the example program, the weights w_k/w_s are taken as D_k/D_s, or the ratio of competitor diameter to subject tree diameter.

A special problem with tree-position indices of competition is the treatment of edge effects or plot-edge bias. As trees become progressively nearer the edge of the plot, so the competition index will become increasingly incomplete. There are several methods of compensating for this which have been reviewed by Monserud & Ek (1974). These include mirroring or wrapping the plot topology, or weighting the competition effect in some way for edge trees. A simple method that appears effective is that due to Bella (1970). This method involves shifting the plot image to obtain a 'virtual plot' around the measured plot so that all trees have competitors over the same distance. The concept is shown in Figure 10. A given subject tree, near the upper left corner, has real trees as competitors where they occur within the plot, and 'virtual trees' outside the plot. The virtual trees are obtained by centering the real plot data on the subject tree. Example 28 demonstrates the algorithm by which this generation of 'virtual competitors' is achieved. The plot is defined as a rectangle extending from coordinates **(xa,ya)** to **(xb,yb)** and with centre at **(xc,yc)** in program lines 5-8. For each real tree, a corresponding virtual tree is generated using the original tree coordinates less the coordinates of the plot centre. The virtual trees are tested to ensure that they are outside the real plot; they are then used as potential competitors (program lines 34-47).

This method of correcting for plot-edge bias has its limitations, as do other techniques. The effect of a large tree on the plot may be replicated onto the virtual plot, giving an abnormally high apparent basal area. Likewise, the influence of gaps may be duplicated. Methods based on mirroring or wrapping the plot near the edge can similarly amplify extremes of high or low basal area. It is doubtful whether there is any ideal solution to this problem, and individual tree competition indices must be assumed to be conditioned by plot size and shape, and the plot-edge correction method used (Vanclay, 1994:61).

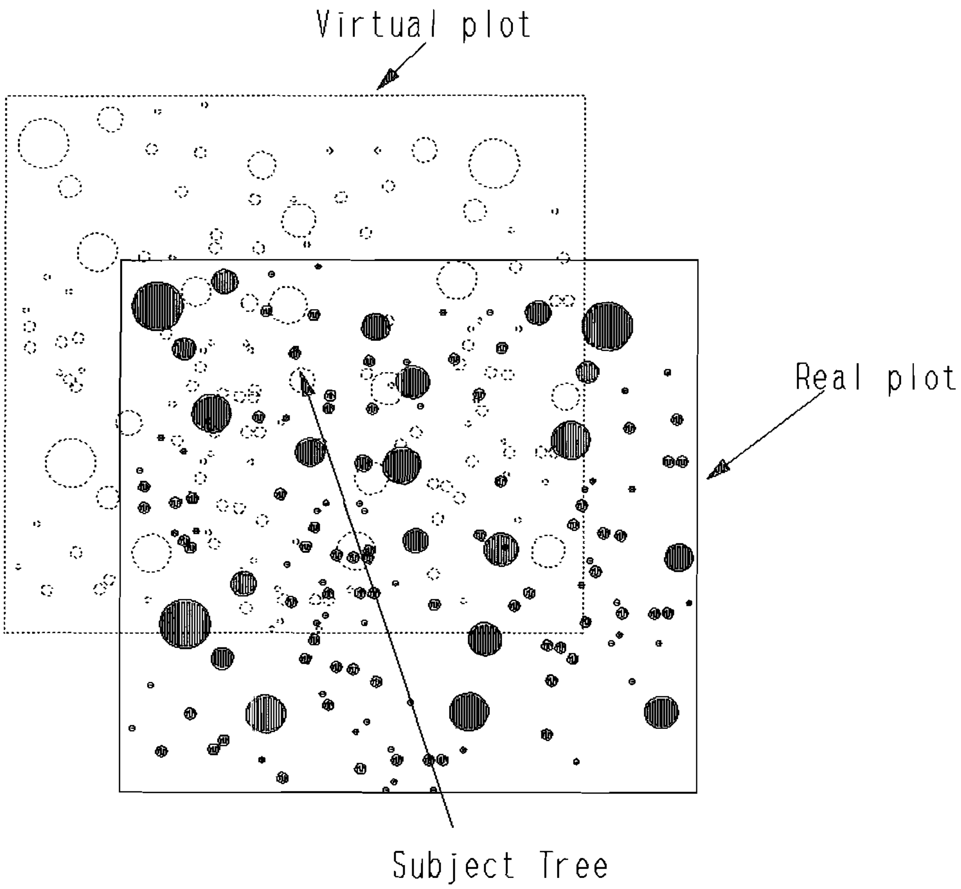


Figure 10 *Using a virtual plot to eliminate edge effects in tree-position competition index estimation*

Program CIOZ listed in Example 28 calculates and fills in a field CIO for a simple PSP database. The database should have a structure that includes PLOTID and TNO as character fields for plot and tree identification, tree coordinates as numeric fields LOCE and LOCN, species code as SPP (either as a character or numeric field), and tree diameter as DIAM. The program is run with the database name supplied as a parameter, viz:

DO CIOZ WITH [*pspfile*]

A second database is needed to define the ratios of tree diameter to competitive influence radius for each species. This requires the fields SPP and RATIO. SPP must have the same form as the SPP field in the PSP file. RATIO is a numeric field.

The program creates an internal list of plots to be processed (**plotlist**) using an SQL SELECT statement at line 10. A loop (lines 11-57) works down this plot list. For each plot, a list of trees is selected from the database (line 13-14). This is linked with the database CIR of species and ratios of diameter to competitive radius. The list is stored in array **tree** which has 5 columns, corresponding to tree number (TNO), coordinates (LOCE,LOCN), tree diameter (DIAM) and competition influence ratio.

Having produced this tree list, the program then works through it, treating each

```
* calculates Competitive Influence Overlap Zones.  Written for FoxPro 2.6
parameters pspdata
set talk off
* define plot dimensions (here 50 x 50 m)
store 0 to xa,ya
store 50 to xb,yb
xc=(xb-xa)/2
yc=(yb-ya)/2
* create array 'plotlist' containing list of Plot IDs to process and work through list
select distinct PLOTID from (pspdata) into array plotlist
for np=1 to alen(plotlist)
  * get tree data needed for CIO calculation (Tree ID, coordinates, CI radius)
  select all TNO,LOCE,LOCN,DIAM,DIAM*RATIO from (pspdata) t,cir z ;
         into array tree where PLOTID=plotlist(np) and t.SPP=z.SPP
  nt=alen(tree,1)
  dimension cio(nt)
  * work through trees on plot calculating CIO for each
  for s=1 to nt
    cio(s)=0
    r1=tree(s,5)
    xs=tree(s,2)
    ys=tree(s,3)
    * add CIO for real competitors
    for k=1 to nt
      dt=sqrt((tree(k,2)-xs)^2+(tree(k,3)-ys)^2)
      r2=tree(k,5)
      if k<>s .and. dt<(r1+r2)
        cio(s)=cio(s)+overlap(r1,r2,dt)*weight(tree(k,4))
      endif
    next
    * add CIO for virtual competitors (to compensate for edge effects)
    x=xs-xc
    y=ys-yc
    for k=1 to nt
      xv=tree(k,2)+x
      yv=tree(k,3)+y
      if xv<xa .or. xv>xb .or. yv<ya .or. yv>yb
        dt=sqrt((xv-xs)^2+(yv-ys)^2)
        r2=tree(k,5)
        if dt<(r1+r2)
          cio(s)=cio(s)+overlap(r1,r2,dt)*weight(tree(k,4))
        endif
      endif
    next
    cio(s)=cio(s)/(pi()*r1^2*weight(tree(s,4)))
  next
  * write CIO values back to original file  (assumed indexed on plotid+tno)
  close databases
  use (pspdata) in 0 order tag treeid
  for t=1 to nt
    seek plotlist(np)+tree(t,1)
    replace CIO with iif(cio(t)>0.00005,cio(t),0)
  next
next
return
```

Example 28 *Program CIOZ to calculate CIO competition index from tree coordinate and size data*

```
1   function overlap
2   parameters r1,r2,d
3   * gives overlap area of two circles of radius r1,r2 and distance d
4   do case
5   case d>=(r1+r2)
6     * circles don't overlap
7     olap=0
8   case d<=abs(r1-r2)
9     * One tree wholly inside influence zone of other.  Use area of smaller.
10    olap=pi()*min(r1,r2)^2
11  otherwise
12    * circles partially overlap (note: angles in radians)
13    r1q=r1^2
14    r2q=r2^2
15    dq=d^2
16    theta=acos((r1q-r2q+dq)/(2*d*r1))
17    phi=acos((r2q-r1q+dq)/(2*d*r2))
18    olap=r1q*(theta-sin(theta)*cos(theta))+r2q*(phi-sin(phi)*cos(phi))
19  endcase
20  return olap

21  function weight
22  parameters diam
23  * weights crowns as a function of diameter - used to compensate for
24  * height differences
25  return diam
```

Example 29 *Function to calculate overlap areas, called by program CIOZ*

tree in turn as a subject tree for calculation of the CIO index. This is done in the loop from lines 18-49. The real competitors are calculated first, in the loop from lines 25-32. The distance **dt** between the subject tree **s** and a competitor **k** is calculated at line 26. The radii of the subject tree and competitor are stored in **r1** and **r2** respectively. The distance is checked at line 28 to ensure that the trees do overlap, and also that the subject tree is not a competitor to itself (ie. **s**≠**k**). If the tree is a competitor, then the CIO is calculated with the **overlap** function (see Example 29). Virtual competitors, lying outside the real plot, are analysed in lines 34-47 using similar calculations, but with shifted tree coordinates (lines 37-38) and a test at line 39 to restrict the virtual competitors to those outside the plot.

Lines 50-53 of the program write the calculated CIO values for each tree back out to the PSP file. This must have been previously indexed with a command:

```
INDEX ON PLOTID+TNO TAG TREEID
```

and must have the field CIO in its structure. The IIF() function on the REPLACE command at line 52 is designed to avoid the problem of underflow when writing to the CIO field with a fixed number of decimal places. It sets CIO to zero if the computed value will result in a loss of data (and hence an error message) when truncated.

The function to calculate the overlap area is shown in Example 29. The CASE statement in this function selects one of three possible situations:

(i) *No overlap*: This occurs when the distance between trees **d** is greater than the sum of their radii (**r1** + **r2**). The value of overlap area (**olap**) is zero.

(ii) *One tree wholly inside the other*: Here the area of overlap is the area of the smaller circle, which may be either the subject tree under a larger competitor, or a competitor under a large subject tree.

(iii) *Partial overlap:* This situation is described by equations {2} to {4}, which are given in program lines 13-18 of Example 29.

The **weight** function is included for completeness. In this example it is trivial, and simply returns the diameter passed to it as a parameter. It could however use a function of diameter (such as D^2), or, with some program modifications, be made sensitive to a species-dependent feature such as foliage density.

This introduction to the calculation of tree position competition indices should enable the reader to make a start on this complex subject. It will be clear that there are many ramifications and variations possible. As Vanclay (1994:61-62) notes, the correlation of tree increment with these complex indices may well be no better than that of much simpler indices such as stand basal area. The principle objective of this section in the context of this manual has been to provide the reader with the computational tools needed to compare the efficacy of tree position and position-independent indices.

3 DIAMETER CLASS PROJECTION METHODS

3.1 Definition and principles

3.1.1 *Definition and domain of application*

Diameter class projection methods represent the oldest class of mathematical models developed for growth projection in tropical forest, with their application beginning with Brandis in Burma in 1856 (Brasnett, 1953). They have been widely applied to the problem of selection forest management generally, with numerous examples in the literature and in standard textbooks (eg. Husch et al., 1982).

The basic concept of a diameter class projection model (DCPM) is that the forest is represented as a stand table of tree numbers classified by diameter classes. The change in the stand table is calculated over an interval of perhaps 5-10 year using periodic increment data. The revised table is then used as the starting point to repeat the calculations. In this way, increment, mortality and ingrowth observations made from PSPs over relatively short periods (less than 10 years) may be used to estimate growth over a full felling cycle or rotation, which may be 40 years or more.

In their simplest form, DCPMs can use published inventory figures and mean increment and mortality rates, together with a standard spreadsheet program such as Lotus 1-2-3, to give an idea of increment and sustainable harvest. Such methods are useful for basic forest management, feasibility studies and project appraisals. More sophisticated developments of the basic approach comprise complete tropical forest simulation models such as those of Korsgaard (1992, 1994) and Alder (1990).

However, DCPMs have some serious inherent weaknesses, especially when applied to mixed tropical forests. In their basic form they do not represent either the effect of stand density or the consequences of spatial heterogeneity. Matrix algebra formulations, although elegant in theory, become unwieldy and altogether impracticable when dealing with a large number of species and a limited base of growth data. Elaborations to allow for stand density, site quality, and local forest heterogeneity are possible, but the result is likely to be less efficient than an equivalent cohort model.

Therefore the author would recommend that DCPMs are used for preliminary or short-term analysis of yield, and are formulated in relatively simple terms. For more refined, accurate, and process-related modelling of tropical mixed forest, the cohort modelling approach is more suitable.

3.1.2 *The stand table as a description of the forest*

The stand table is a basic tool in forestry. Forest inventory reports for tropical forests typically show tables arranged with diameter classes at the top of the page and species down the left, with stem numbers, basal areas, or volumes given in the table. Example 30 shows some summarised results from three forest inventories. In each case, results are shown for all species, for a typical common and ubiquitous species of the area, and for a species of high commercial importance. The forests represent three different management situations. The Chiquibul results relate to an area of forest that suffered severe damage during Hurricane Hattie in 1961. Tinte Bepo Forest Reserve in Ghana has been subject to continuous selection logging under a polycyclic felling system since 1954, with the primary emphasis on species such as Khaya Mahogany, but with some extraction also of the common species Wawa (*Triplochiton scleroxylon*). The data from Budongo Forest in Uganda relates to the compartments designated as Nature Reserves, which have never been exploited and are not known to have suffered any major catastrophe. The stockings in all tables are shown in trees/km^2.

```
(i) Chiquibul, Belize 1969 : Severe hurricane damage, 1961

Diameter classes               20-40  40-60  60-80 80-100  100+
----------------------------------------------------------------
All species                     8731   1214    335    106    42
Common/Ubiquitous (Nargusta)      84     45     43     29    16
Mahogany (Swietenia)              20      6      2      1
(Johnson & Chaffey, 1973)

(ii) Tinte Bepo, Ghana 1987 : Controlled commercial logging from 1954

Diameter classes               20-40  40-60  60-80 80-100 100-120  120+
-----------------------------------------------------------------------
All species                    11554   1947    993    408     183    58
Common/Ubiquitous (Wawa)         232    146    164    136      64    14
Mahogany (Khaya)                   7      4      4      4
(Owen, 1987)

(iii) Budongo, Uganda 1990 : Unexploited mixed forest

Diameter classes              10-30  30-50  50-70  70-90  90-110 110+
---------------------------------------------------------------------
All species                    31220   5020   1232    876     464   320
Common/Ubiquitous (Celtis)     11400   2480    228     24      12     8
Mahogany (Khaya)                 220            32     24      32    56
(Alder, 1991)
```

Example 30 *Stand tables from some selected inventories in mixed tropical forest*

Simple stand tables can be formed from tree measurements on permanent or temporary plots using a program similar to that shown in Example 20 on page 44. However, a full forest inventory program is usually very much more complex. Apart from the need to produce sampling error statistics, sampling designs usually involve tiered plots of different size for different diameter categories, and sometimes species groups.

These stand tables are presented in terms of tree numbers. Clearly for management purposes, tables of volumes or basal area by size classes may be required.

However for growth projection purposes, the tree numbers table is the most useful. Numbers are presented per km^2, rather than per ha, in order to give a meaningful picture at the low levels of stocking that occur for individual species. It is harder to envisage the significance of a stocking of 0.01 tree/ha than 1 tree/km^2, although they are the same statistic.

A characteristic feature of the stand tables of mixed-species, uneven-aged forests is the progressive decrease in stem numbers from the smaller to the larger diameter classes. This is invariably seen when large areas and groups of species are considered, as the for the 'all species' lines in Example 30.

In temperate-zone selection forests with only a few species, the progressive decrease in stem numbers with size is likely to be true for all except rare or minor components of the stand. In mixed tropical forests, with typically 200-500 species being observed, diameter distributions for particular species frequently occur which show a deficit of smaller trees or a preponderance of larger ones. In Example 30 it can be seen that in cases (i) and (ii), even the common species, Nargusta (*Terminalia amazonica*) and Wawa (*Triplochiton scleroxylon*) show such atypical distributions.

Inventory		<-----	Diameter	Class	------>			Mean
		1	2	3	4	5	6	Q
Chiquibul Forest	N/km^2	8731	1214	335	106	42		
Belize 1969	Q-ratio		7.19	3.62	3.16	2.52		4.12
Tinte Bepo Forest	N/km^2	11554	1947	993	408	183	58	
Ghana 1987	Q-ratio		5.93	1.96	2.43	2.23	3.16	3.14
Budongo Forest	N/km^2	31220	5020	1232	876	464	320	
Uganda 1990	Q-ratio		6.22	4.07	1.41	1.89	1.45	3.01

Example 31 *De Liocourt ratios for 3 tropical forest inventories*

Because of the large-scale differences for stem numbers that occur between the smallest and largest diameter classes, it is usually convenient to plot diameter distributions on semi-logarithmic axes as shown in Figure 11. This makes differences between distributions much clearer. A straight line fit on such axes corresponds to an exponential distribution for the data, as discussed in section 3.1.3.

The stand table can also be plotted in terms of basal area per diameter class versus diameter. Korsgaard (1992) uses this method to present results from the STANDPRO model and commends its advantages. Alder (1994) adopts the same type of graph for the CAFOGROM model. However, the shape of the basal area/diameter curve is less familiar to foresters than the stem numbers/diameter curve, and the diagnostic implications for stand management are less recognisable.

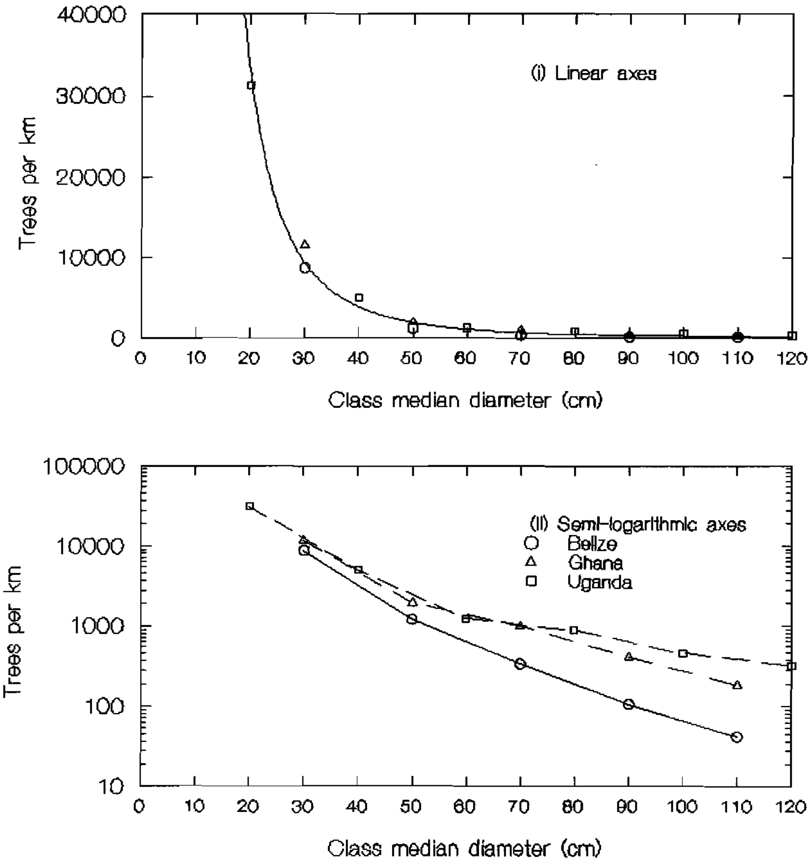


Figure 11 *Semi-logarithmic axes for plotting stand table data*

3.1.3 The De Liocourt Ratio and the J-shaped curve

De Liocourt (1898) noted that the numbers in the successive diameter classes of a the stand table for an uneven-aged forest tend to form a geometric series, such that:

$$Q = N_{k-1}/N_k \qquad \{7\}$$

where N_k are the number of trees in size class k, N_{k-1} are the numbers in the preceding class, and Q is a constant.

The ratio Q between classes is usually termed the De Liocourt Quotient. This is calculated in Example 31 for the 'all species' data from the three inventories in Example 30 using a simple spreadsheet. It can be seen that Q value tends to be in the range 2-3 for the larger diameter classes, and higher for the small classes. The Q value depends directly on the class width. If smaller classes are used, lower ratios will result, whilst with wider classes, higher ratios occur.

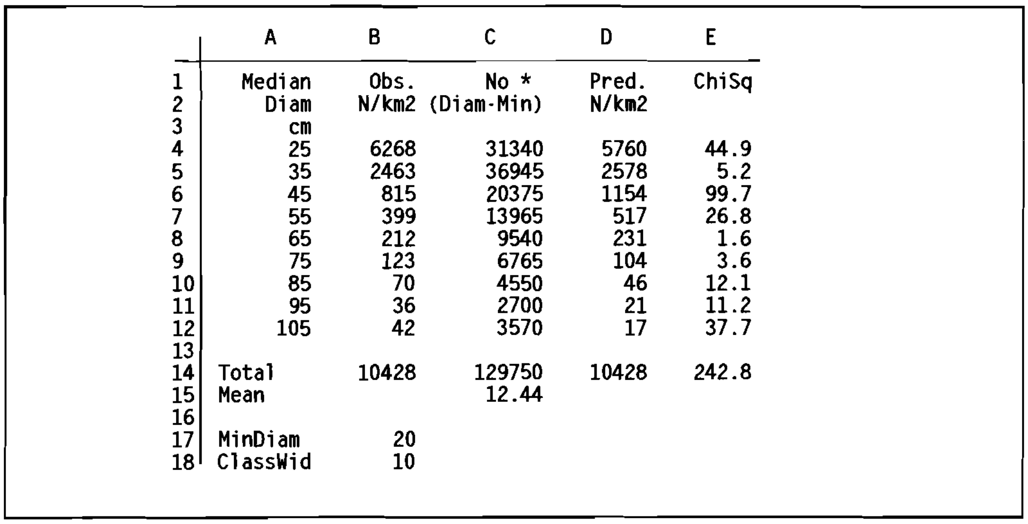
Meyer (1933) showed that a diameter class distribution with constant Q values was mathematically equivalent to an exponential distribution. The exponential distribution for tree numbers with respect to diameter can be expressed in its integral form as:

$$N_d = N_T.(1\text{-}\exp(-\alpha.(d\text{-}d_0)) \qquad \{8\}$$

where N_d is the total number of trees up to diameter d, N_T is the total number of trees, d_0 is the lower limit of the smallest diameter class, and α is an empirical constant. An unbiased estimate of the parameter α is given by the reciprocal of the mean of $(d\text{-}d_0)$, which for classified or grouped data is given by:

$$1/\alpha = \Sigma[(d_k\text{-}d_0).N_k]/\Sigma N_k \qquad \{9\}$$

where d_k is the median diameter of the k'th class, d_0 is the lower limit of the smallest diameter class, and N_k is the number of trees in the k'th class. Example 32 demonstrates these calculations and shows how the predicted distribution can be fitted, using data from the Belize Chiquibul inventory classified by 10 cm classes. Column A in the spreadsheet contains the diameter classes, which are calculated from the values 'MinDiam' and 'ClassWid' given in cells B17 and B18. Column B has in it the observed numbers of trees, entered as data. Column C gives the products (Ax-B17)*Bx, where x is the row number. The sum of these products, in cell C14, divided by the sum of the observed stem numbers, in cell B14, gives the mean relative to the minimum diameter. The reciprocal of this is the exponential distribution parameter α. Predicted values from the distribution are calculated in column D.



	A	B	C	D	E
1	Median	Obs.	No *	Pred.	ChiSq
2	Diam	N/km2	(Diam-Min)	N/km2	
3	cm				
4	25	6268	31340	5760	44.9
5	35	2463	36945	2578	5.2
6	45	815	20375	1154	99.7
7	55	399	13965	517	26.8
8	65	212	9540	231	1.6
9	75	123	6765	104	3.6
10	85	70	4550	46	12.1
11	95	36	2700	21	11.2
12	105	42	3570	17	37.7
13					
14	Total	10428	129750	10428	242.8
15	Mean		12.44		
16					
17	MinDiam	20			
18	ClassWid	10			

Example 32 *Spreadsheet showing fitting of exponential distribution to diameter class data*

The predicted values in column D are calculated from equation {8} by noting that:

$$N_k = N_{d\text{-}w/2} - N_{d+w/2} \qquad \{10\}$$

where d is the class median, and w is the class width of the k'th class. Column E gives values to calculate the Chi² statistic to test goodness of fit to the distribution. They are values of $(f-F)^2/F$, where f is the observed frequency, column B, and F is the expected frequency, column D. The Chi² statistic is the sum of these, given in cell E14. The critical value of Chi² with 7 degrees of freedom is 14.07 at P=0.95, leading to the conclusion that the observed distribution differs significantly from the fitted exponential model.

Leak (1965) provides additional information about the exponential distribution, its formulae and transformations as applied to diameter distributions. Meyer (1952) discusses the concept of a balanced diameter distribution as one which conforms closely to the exponential model, and which may be selection-cut on a sustained basis to maintain the same distribution, with felling equal to growth.

In tropical forests, as noted in the examples above, the Q value does not appear to be constant, but tends to decline from high values at small size classes to lower values for the largest tree. In this case, the Weibull function is more appropriate (Leak, 1970). Mathematically, this is similar in form to the exponential function, with an additional coefficient ß:

$$N_d = N_0.(1-\exp(-\alpha.d^{\beta})) \qquad \{11\}$$

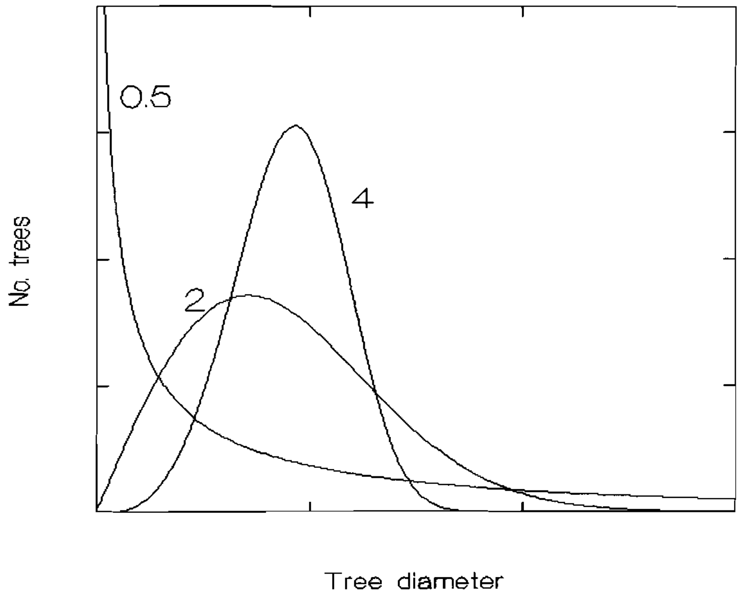


Figure 12 ***Shape of the Weibull distribution with different ß parameters***

Bailey & Dell (1973) discuss the features of the Weibull function as it relates to diameter distributions. It will be noted that when ß=1 then the Weibull and Exponential functions are the same. With values of ß<1, then the curve will be a reverse J-shape, but will be steeper than a true exponential curve. If ß>1, then the Weibull becomes a unimodal distribution. For values of ß between 1 and 3.6, the distribution is right-skewed. At 3.6, it is approximately symmetric and bell-shaped, superficially similar to a normal distribution. For higher values of ß, the distribution is right-skewed. Some examples of shape are shown in Figure 12. The α parameter does not effect the shape, but influences the scale of the function.

Equation {11} can be modified to give the number of trees in a specified size class k from d_{k-1} to d_k as:

$$N_k = N_0.[\exp(-\alpha.d_k^{\beta})-\exp(-\alpha.d_{k-1}^{\beta})] \qquad \{12\}$$

Simple parameter estimation methods for determining the coefficients of the Weibull distribution from data are discussed in section 4.2.7, page 127.

Early forest biometricians appear to have been much impressed by the relatively constant values of Q found in temperate-zone selection forests, and the linearity of the diameter distribution when plotted on semi-logarithmic axes, attributing a normative significance to it (Meyer, 1952). Assmann (1970:450) states that the coefficients N_0 and α are important indicators of the structure and age conditions of selection stands. However, in mixed tropical stands, the premise of a constant Q-value does not generally hold well, and the focus of management is usually on a small group of commercially valuable species which are likely to have a quite irregular distribution. As Oliver & Larson (1990:297) note, the Q-ratio and mathematical distributions such as the exponential and Weibull should be regarded as descriptive, rather than as desirable ends to be attained through forest management.

3.1.4 *Ingrowth, outgrowth, mortality and harvest*

The change in a stand table over a period of time can be described in terms of ingrowth, outgrowth, mortality, and harvest. These are defined as follows:

> *Ingrowth* are the trees which grow into a diameter class over a given period. Trees growing into the lowest measured diameter class are termed *Recruitment*, or sometimes *External ingrowth*.
>
> *Outgrowth* are those trees which grow out of a particular diameter class over a period. Ingrowth into a class corresponds to outgrowth from the preceding class.
>
> *Mortality* are trees which die during a growth period. The causes may be internal to the stand, as a result of suppression, shading, or age; or external, as a result of catastrophes such as tropical storms. It is not usually possible to strictly separate the two, as age or disease predisposes trees to catastrophic breakage.
>
> *Harvest* implies trees which are removed during logging over a period.

Mathematically, the process of diameter class projection can be defined as:

$$N_{k,t+1} = N_{k,t} + I_k - O_k - M_k - H_k \qquad \{13\}$$

where:

$N_{k,t+1}$ is the number of trees in the k'th class at period t+1
$N_{k,t}$ is the number of trees in the k'th class at period t
I_k is the ingrowth into the k'th class during the period
O_k is the outgrowth from the k'th class during the period
M_k is the mortality from the k'th class during the period
H_k are the trees harvested from the k'th class during the period.

The general assumption of diameter class projection is that values of ingrowth, outgrowth and mortality derived from PSP data measured over intervals of 5-10 years or so can be applied repetitively to obtain new estimates of a stand table over successive periods.

It will be apparent that ingrowth into class k will be numerically equal to outgrowth from class k-1. If ingrowth is therefore restricted to imply 'external ingrowth' only, symbolized by R (recruitment), then equation 3.5 can also be written as:

$$N_{k,t+1} = N_{k,t} + R_k + O_{k-1} - O_k - M_k - H_k \qquad \{14\}$$

where R_k are the numbers of trees appearing in diameter class k as a result of external ingrowth.

Ideally, if simple diameter class projection is being applied as a method of growth forecasting, then the quantities R, O, M and H should be compiled directly from PSP data. This minimizes bias or error arising from the introduction of unverified assumptions into the model. Some practical problems will emerge if this approach is tried. Recruitment may occur into classes other than the lowest; whilst outgrowth may similarly involve trees skipping more than one class over a period. These problems can be reduced by using broad diameter classes (say 20 cm) and by using shorter growth periods (not more than 5 years).

Compiling data for rare species may also involve ingrowth, outgrowth or mortality being either not represented for a particular diameter class, or represented only by 1 or 2 trees, giving a very unreliable estimate. This problem can be minimized by grouping species, and also by using broader diameter classes.

Ingrowth/outgrowth, mortality, and harvest may also be defined in terms of percentages of current tree numbers. In this case, using lower case letters o,m,h to denote recruitment, outgrowth, mortality, and harvest respectively as proportions of the existing stocking in a class, then we have:

$$N_{k,t+1} = R_k + N_{k-1,t}.o_{k-1} + N_{k,t}.(1 - o_k - m_k - h_k) \qquad \{15\}$$

This formulation is likely to be more convenient when developing a diameter class projection model using values of mortality and growth culled from other research. Outgrowth rates, as will be seen, can be estimated from mean diameter increments within a class.

3.1.5 *Matrix algebra formulations*

Equation {15} can be reformulated using matrix algebra in a way that is both concise, and allows growth of trees through several diameter classes in one time period. The basic matrix formulation can be stated as:

$$\mathbf{N}_{t+1} = \mathbf{G}.\mathbf{N}_t \qquad \{16\}$$

where:

$\mathbf{N}_t$ is a column vector* whose elements are the stem numbers in each diameter class at time t. It may be written as:

$$\mathbf{N}_t = [\, n_1 \;\; n_2 \;\; n_3 \; \ldots \; n_k \; \ldots \; n_m \,]' \qquad \{17\}$$

The elements n_k are identical to the scalar values N_k in equation {15}. There are m diameter classes altogether.

$\mathbf{N}_{t+1}$ is a column vector of stem numbers by diameter classes one time period later. Its definition is as in {17}.

$\mathbf{G}$ is a square matrix of order m known as the transition matrix. Each element g_{ij} defines the proportion of stems which grow from the i'th diameter class to the j'th diameter class during a time period. The elements of **G** correspond to components of equation {15} for trees which move exactly one class or remain in the same class. However, they can also allow growth of trees across several classes, or even shrinkage of trees.

Equation {16} works because of the way the operation of matrix multiplication is defined. Each element of $\mathbf{N}_{t+1}$ will be given as:

$$n_{j,t+1} = \Sigma^{m}{}_{k=1} n_{k,t}.g_{kj} \qquad \{18\}$$

This is implicit in the definition of matrix multiplication (see for example Green, 1976:46). The application of matrix algebra to growth modelling in uneven-aged forest stands appears to have evolved from three distinct lines of thought. The models developed by Usher (1966) were derived from animal demographic methods applied to plant populations. Bruner & Moser (1973) used the general

* Readers unfamiliar with the terminology of matrix algebra are referred to basic mathematical texts on the subject. The author has found Green (1976) to be useful.

technique of Markov modelling, whilst other authors (eg. Mengin-Lecreulx, 1990) simply adapt matrix algebra to the 'classical' diameter class projection model defined in equation {15} above.

The use of matrix models in forestry appears to have originated with Usher (1966). He noted that Leslie (Leslie, 1945, 1948) had described a matrix model of animal populations with the following form:

$$\begin{bmatrix} a_{t+1,1} \\ \vdots \\ a_{t+1,i} \\ \vdots \\ a_{t+1,n} \end{bmatrix} = \begin{bmatrix} f_0 & \cdots & f_i & \cdots & f_{n-1} & f_n \\ p_0 & \cdots & \cdots & \cdots & . & . \\ . & \cdots & p_i & \cdots & . & . \\ . & \cdots & \cdots & \cdots & p_{n-1} & . \end{bmatrix} . \begin{bmatrix} a_{t,1} \\ \vdots \\ a_{t,i} \\ \vdots \\ a_{t,n} \end{bmatrix} \qquad \{19\}$$

which can be expressed more compactly in matrix notation as:

$$\mathbf{a}_{t+1} = \mathbf{A}.\mathbf{a}_t \qquad \{20\}$$

The column vectors $\mathbf{a}_t$ and $\mathbf{a}_{t+1}$ are the numbers of animals in each age class at time t and t+1 respectively. The square matrix **A** contains a top row of elements f_0 to f_n which are the fecundities of each age class. In other words, f_i is the number of offspring that will be born to the i'th class during one time interval.

The off-diagonal elements p_i are the probabilities that any individual will survive over a time period.

Usher adapted this model from the context of animal demography to tree growth and regeneration, reformulating equation {19} as:

$$\mathbf{q}_{t+1} = \mathbf{Q}.\mathbf{q}_t \qquad \{21\}$$

Instead of the population vector **a** with elements representing age classes, we have the vector **q** whose elements correspond to tree diameter classes*. The transition matrix **Q** in equation {21} is analogous to **A** in {20}, but with some differences of interpretation. It is constructed of zero elements except for a top row k_i, diagonal elements a_i and off-diagonal elements b_i as shown in equation {22}. The interpretation of these elements in forestry terms is as follows:

- The a_i terms are the proportion of static trees that remain in a given class during any growth period.

- The b_i terms are the proportion of trees in the i'th size class that will grow into the next class up (i+1) during the period.

* In this and the following examples of matrix models developed by different authors, the author's original notation is used. The letter Q here, used by Usher, should not be confused with the De Liocourt Quotient discussed in section 3.1.2.

$$
Q = \begin{bmatrix}
a_0 & k_1 & k_2 & \cdots & k_{n-1} & k_n \\
b_0 & a_1 & \cdot & \cdots & \cdot & \cdot \\
\cdot & b_1 & a_2 & \cdots & \cdot & \cdot \\
\cdot & \cdot & \ddots & \ddots & \cdot & \cdot \\
\cdot & \cdot & \cdot & \ddots & a_{n-1} & \cdot \\
\cdot & \cdot & \cdot & \cdots & b_{n-1} & a_n
\end{bmatrix} \qquad \{22\}
$$

- The k_i terms are the total numbers of ingrowth, or plants added to size class 0, during a period, as a result of trees being harvested from the i'th class. For classes 0 to n-1, these terms can be defined as:

  $$k_i = c_i(\alpha\text{-}1) \qquad \{23\}$$

  and for the n'th class as:

  $$k_n = c_n(\alpha\text{-}a_n) \qquad \{24\}$$

  where α is the proportion of trees left after harvesting, and c_n is the number of ingrowth trees expected to arise in the gap left by a single tree in that class.

Bruner & Moser (1971) developed a model of stand growth for uneven-aged mixed hardwood forest in Wisconsin that is purely Markovian* in its design. The basic model is identical in formulation to equation {16}. The elements of the state vector consisted of the size classes from 8" to 29", 29"+, dead trees, and harvested trees. Permanent sample plot data was used to calculate the transition probabilities, which aggregated for all species. The paper does not indicate either the species mix or the number of plots used to derive the data, but the original transition matrix is reproduced. It appears that annual measurements existed over a 19-year period; the first nine measurements were used to define the transition matrix, which thus had a 9-year time step. The subsequent measurements were used for validation.

This shows that trees can grow one or more 1" class during the time step. Mortality is small but positive for the smallest and largest classes, and zero for the mid-sized trees 19-25". Harvesting also occurs at all sizes, but principally above 22" diameter, where it is about 20% of the class stocking.

The model was used to make forecasts of stand structure over an 18-year period (2 time steps) by squaring the transition matrix. In general with a Markov model, the state of the system after n steps, $\mathbf{t}_n$ can be determined analytically from the initial state $\mathbf{t}_0$ by the matrix equation:

* The nature of Markov models is discussed in Vanclay (1994:29).

$$\mathbf{t}_n = \mathbf{P}^n.\mathbf{t}_o \qquad \{25\}$$

Bruner & Moser's paper included a comparison between the model's projection over 9 years and the observed growth and mortality. The fit appears good with respect to both trees numbers and diameter distribution.

Bruner & Moser appear to have been unaware of Usher's (1966) work, in spite of the close similarities of the approach. Usher's method concentrated to some extent on regeneration, and allowed harvesting only as the same constant proportion in each size class (the a factor in equation {24}). Usher also allowed growth over only a single diameter class. Bruner & Moser's work, on the other hand, explicitly neglected regeneration, but included a flexible method of defining the harvest and mortality rates.

Buongiorno & Michie (1980) extended Usher's formulation of the matrix model by including a separate vector of harvested trees, and by allowing ingrowth to be a density-dependent function of basal area at the previous time period. In matrix notation, the model is:

$$\mathbf{y}_{t+1} = \mathbf{G}(\mathbf{y}_t - \mathbf{h}_t) + \mathbf{c} \qquad \{26\}$$

where:

$\mathbf{y}_t$ is a column vector of tree numbers by size classes in the current growth period.

$\mathbf{y}_{t+1}$ is a column vector of tree numbers by size classes at the next growth period.

G is the transition matrix, comprising the following non-zero elements:

$$G = \begin{bmatrix} d_1 & d_2 & d_3 & \cdots & d_n \\ b_2 & a_2 & . & \cdots & . \\ . & b_3 & a_3 & \cdots & . \\ . & . & \ddots & \ddots & . \\ . & . & . & b_n & a_n \end{bmatrix} \qquad \{27\}$$

$\mathbf{h}_t$ is a column vector of tree numbers harvested during the period.

c is a column vector comprising only a single non zero element:

$$c = \begin{bmatrix} \beta_0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \{28\}$$

The a_i terms define, as in the other matrix models, the proportion of trees that remain in the i'th class. The b_i elements likewise define the proportion of trees in i'th class that will move into the (i+1)'th class during a growth period. The d_i and $ß_0$ elements relate to the formulation of the ingrowth function. It is assumed that the number of trees in the smallest class at the next growth period can be predicted from the equation:

$$y_{1,t+1} = ß_0 + d_1(y_{1,t} - h_{1,t}) + \ldots + d_n(y_{n,t} - h_{n,t}) \qquad \{29\}$$

where:

$$d_1 = a_1 + ß_1 B_1 + ß_2 \qquad \{30\}$$

$$d_i = ß_1 B_i + ß_2 \qquad \{31\}$$

The B_i are the mean basal areas of each size class, whilst the $ß_i$ are empirical coefficients. When the matrix multiplication is worked out, equation {29} resolves to an equation to predict ingrowth into the smallest class with the form:

$$I_t = ß_0 + ß_1 \Sigma\{B_i(y_{i,t}-h_{i,t})\} + ß_2 \Sigma(y_{i,t} - h_{i,t}) \qquad \{32\}$$

In other words, ingrowth is a linear function of stand basal area and the total number of trees. The $ß_1$ term is likely to be negative, with lower ingrowth at higher stand densities due to shading; the $ß_2$ term should be positive, indicating that that there will be more seedlings developing as more gaps are formed from dead or felled trees.

Buongiorno & Michie tested this model with data from mixed hardwood forests in Wisconsin and Michigan, dominated by Sugar Maple. They found that over long periods of simulated time, the stocking tended to oscillate in long cycles of some 200 years, with the diameter distribution tending to be U-shaped, with abundant large trees and a deficit of intermediate sizes. This is interesting as such diameter distributions may be observed in undisturbed tropical high forest (Alder, 1991).

The paper also explores the conditions and effects for a sustainable yield defined such that the stand vector **y** at the end of a felling cycle is the same as at the beginning. It is possible to solve for the yield that will satisfy this condition using linear algebra. The result is a vector of tree numbers to be removed, with each element being a size class.

In later papers on the same theme, Michie & Buongiorno (1984) gave more details of how the elements of the transition matrix and the empirical coefficients could be calculated from permanent sample plot data. Buongiorno & Hsien-Chi (1990) discuss how the basic model in equation {26} can be used in a linear program that maximizes the value of the harvest, subject to the sustained yield condition that the stand vector should be constant at the end of successive felling cyles. From this, both the optimum harvest and the cutting cycle can be determined. The modelling approach has been applied to forest sector projections in Nigeria (World Bank, 1992)

Mengin-Lecreulx (1990) reports on a matrix model developed for mixed tropical forest in Yapo, Côte d'Ivoire. The basic model is:

$$\mathbf{x}_{t+2} = \mathrm{v}.\mathbf{P}.\mathbf{x}_t+\mathbf{R} \qquad \{33\}$$

where:

$\mathbf{x}_t$ is a column vector of diameter classes in year t. Each class is 5 cm wide, and the model had 25 classes in all. The basic time period was 2 years, hence the vector $\mathbf{x}_{t+2}$ gives the stand 2 years later, after a single growth step in the model.

v is the survival over each growth step, given as a constant fraction irrespective of size class.

P is a growth matrix giving the transition probabilities. As with the matrix Q in equation {21}, only the a_i diagonal elements and b_i sub-diagonal elements are non-zero, and define the proportions of static trees and those which move into the next higher class in each 2-year step.

R is a column vector whose elements are zero except for the first, after the manner of the **c** vector in equation {3.19}. This first element represents a constant rate of recruitment into the smallest diameter class.

The data for the plots was derived from experimental plots at Irobo, Côte d'Ivoire. Different growth, mortality and recruitment functions were established for 30 species and for thinned and unthinned forest, and used to project the growth of the forest at Yapo.

These examples illustrate some of the various formulations of matrix models of stand growth. All are essentially related to the simpler diameter class projection model of equation {15}. Matrix algebra provides a compact notation for discussing the basis of a model. However, in practice, as will be seen, there are significant difficulties in applying matrix models to tropical mixed forests due to:

- The large number of species. Even with grouping of data for species of similar habit, there may be 40-50 groups, each of which requires its own transition matrix, all of whose elements have to be defined as model parameters from data.
- The inefficiency of a matrix representation. In the examples discussed, most of the elements of the transition matrix are zero. If this algebraic

approach is translated directly into a computer program, then the majority of the memory is used storing zeroes, and most of the computer time looping through empty portions of matrices.

Matrix models also suffer from the general deficiency of diameter class projection models, arising from the lack of sensitivity to density-dependent interactions.

3.2 Construction of diameter class projection models

3.2.1 *A spreadsheet model based on mean class increments*

For various reasons, it is often difficult to compile outgrowth rates directly from PSP data. In particular:

- Not all diameter classes may be represented, making it impossible to estimate outgrowth for some classes.
- The data may be sparse or very unevenly distributed between classes, requiring some method of smoothing be applied to obtain reasonable projections.
- Only processed or published increment rate data may be available.
- Data may only be available from tree increment plots, not from PSPs with complete measurement of the stocking.

For these reasons, it is usually necessary to estimate outgrowth from mean increment for a class. The mean increment may itself be estimated directly from class data (using for example, program XTAB on page 44), or via a growth function of increment on diameter.

Given an estimated mean increment for a diameter class of i cm/yr, a class width of w cm, and a projection period of t years, then the proportion of trees moving to the next class as outgrowth (o) may be calculated from:

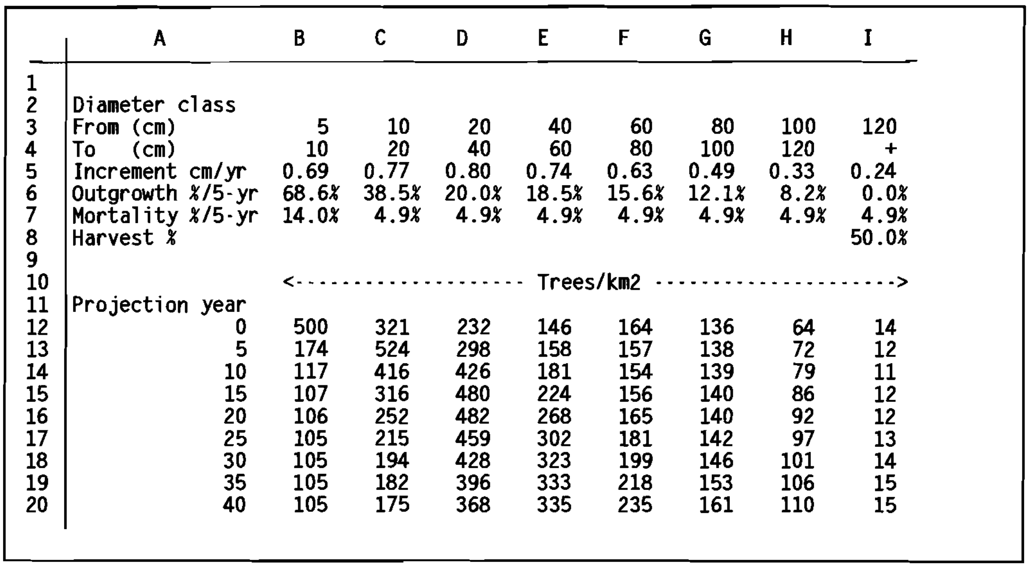
$$o = t.i/w \qquad \{34\}$$

Equation {34} assumes that:

- Trees are uniformly distributed within the class.
- All trees in the class grow at the mean rate.

We may call these the UDMI (Uniform Distribution, Mean Increment) assumptions. These assumptions do not hold well if the classes are broad, but become reasonable approximations with class widths of 5 cms or less.

The construction of a complete diameter class projection model based on the UDMI assumptions and equation {15} can be illustrated using some published data for *Triplochiton scleroxylon* (Wawa) from Ghana. The spreadsheet in Example 33 shows the complete model. The inventory data for the original stand table is taken from Owen (1987). The growth and mortality figures are based on Alder (1992).



	A	B	C	D	E	F	G	H	I
1									
2	Diameter class								
3	From (cm)	5	10	20	40	60	80	100	120
4	To (cm)	10	20	40	60	80	100	120	+
5	Increment cm/yr	0.69	0.77	0.80	0.74	0.63	0.49	0.33	0.24
6	Outgrowth %/5-yr	68.6%	38.5%	20.0%	18.5%	15.6%	12.1%	8.2%	0.0%
7	Mortality %/5-yr	14.0%	4.9%	4.9%	4.9%	4.9%	4.9%	4.9%	4.9%
8	Harvest %								50.0%
9									
10		<-------------------- Trees/km2 -------------------->							
11	Projection year								
12	0	500	321	232	146	164	136	64	14
13	5	174	524	298	158	157	138	72	12
14	10	117	416	426	181	154	139	79	11
15	15	107	316	480	224	156	140	86	12
16	20	106	252	482	268	165	140	92	12
17	25	105	215	459	302	181	142	97	13
18	30	105	194	428	323	199	146	101	14
19	35	105	182	396	333	218	153	106	15
20	40	105	175	368	335	235	161	110	15

Example 33 *A simple stand projection model in spreadsheet format*

Rows 3 to 8 of the spreadsheet establish the basic parameters of the model. Diameter class upper and lower limits are in rows 3 and 4. Row 5 contains increment values, which have been smoothed by fitting a function to the original data. Row 6 contains percentage outgrowth calculated using equation {34}. The spreadsheet formula for cell B6 for example is:

B6 = 5 * B5/(B4-B3)

By copying B6 to cells C6 to I6, this formula will be automatically adjusted to calculate the outgrowth % for each class.

Row 7 contains 5-year mortality percentages entered directly as data. Row 8 contains the percentage of trees in each class to be harvested. Harvesting has been restricted in this example to the largest diameter class ($\geq$120 cm), where 50% of the trees are removed.

The inventory data is entered for projection year 0 in row 12. Thus for example, cell B12 contains the stocking per km^2 for trees of 5-10 cm diameter.

Row 13 shows the projected stand table after 5 years. The formula for cell B13 is:

+B12*(1-B$6-B$7)+87

In other words, the stocking in the smallest, 5-10 cm class after 5 years is the stocking at year zero (B12) less the percentage outgrowth (B$6) and the percentage mortality (B$7) for that size class, plus a constant ingrowth (estimated from PSP data) of 87 stems/km^2 over the 5-year period. The $ signs in the cell references denote that the following row numbers (6 and 7) are held constant when this formula is copied down to subsequent cells B14 to B20[*].

Cell C13 is defined as:

```
+B12*B$6+C12*(1-C$6-C$7)
```

which is anologous to equation {15} with regeneration R_k of zero. The term B12*B$6 is the outgrowth from the preceding diameter class. The remaining terms deduct outgrowth to the next higher class, and periodic mortality, from the current stocking. Cells D13 to I13 are the same as cell C13, but with the column letters adjusted correspondingly.

The projections for periods beyond year 5 are made using the Copy operation in the spreadsheet to copy down row 13 for as many periods as required. Provided absolute and relative cell addresses have been correctly defined, the results will be as shown.

By adjusting the harvest value in cell I8, the system can be studied to determine that percentage cut which will result in a sustained yield in that size class. With less than 50% harvest, stock tends to accrue; with a greater percentage cut, the stock in the last class declines.

The spreadsheet approach is easy to apply when working with a single species. It becomes cumbersome with many species. Example 34 shows a BASIC subroutine which carries out the same calculations as the spreadsheet example. It has however a number of advantages. It can work with any number of species at a time, treating each as an independent table. The diameter and mortality functions called (GROWTH and DEATH respectively) are flexible and could be density-dependent functions. Ingrowth is similarly flexible and could be time or density dependent.

The routine is called with the stand table ST as a parameter. This is an array of two dimensions, the first for species, the second for size classes. The indexes J% and K% are suffixed with % signs to indicate that they are integer values.

The GROWTH function returns the proportion of moving trees. These may be calculated directly from increment using equation {3.25}, or by some more sophisticated method, such as those described in section 4.2. The DEATH function gives a mortality rate. Both functions give values per time period, not per annum. INGROWTH returns the actual number of ingrowth trees into the smallest class.

When the routine STPROJ returns to the calling program, the ST table has been

[*] The $ operator to indicate absolute cell references is a Lotus 1-2-3 notation. Other spreadsheets may use alternative methods of denoting absolute as opposed to relative cell addresses.

```
SUB stproj (st())
'Updates a stand table of tree diameters by species using simple
'stand projection
' get array bounds. lower bound should be 1.
nsp% = UBOUND(st, 1)
ndc% = UBOUND(st, 2)
DIM st0(ndc%)
FOR j% = 1 TO nsp%
  'keep old stand table as st0 to avoid overlap while updating
  FOR k% = 1 TO ndc%
    st0(k%) = st(j%, k%)
  NEXT k%
  'get first class update using ingrowth function
  st(j%, k%) = ingrowth(j%) + st0(k%) * (1 - growth(j%, k%) - death(j%, k%))
  'get updates to other classes
  FOR k% = 2 TO ndc%
    st(j%,k%) = st0(k%-1)*growth(j%,k%-1) + st0(k%)*(1 - growth(j%,k%) - death(k%,j%))
  NEXT k%
NEXT j%
END SUB
```

Example 34 *A BASIC subroutine for generalized stand table projection*

updated for the growth and mortality over the period.

3.2.2 *A simple transition matrix model*

Transition matrix models based on equation {16} are simple to construct if a large amount of data is available. They are difficult to build with sparse data, when the transition matrix will have many zero elements and will be poorly defined.

The matrix model can be defined as:

$$\mathbf{N}_{t+1} = \mathbf{R} + \mathbf{G}.\mathbf{N}_t \qquad \{35\}$$

This is identical to the definition of equation {16} but with an additional column vector, **R**. Each element of **R** gives the numbers of trees growing into the corresponding diameter class as recruitment over a fixed period. **G** is the matrix of transitions, with each element g_{ij} defining the number of trees which move to class i from class j. The column totals will normally be less than one. In other words:

$$\sum_{i=1}^{m} g_{ij} \leq 1 \qquad \{36\}$$

The summation gives the total survival over the period. This type of formulation differs from a pure Markov model such as that of Bruner & Moser (1971) discussed on page 73. In a Markov model, states would need to be defined for dead and harvested trees, so that column totals would always equals 1.

Program TRANSIT illustrated in Example 35 is designed to build a transition matrix

```
' Transition matrix compilation program
CLS
PRINT "TRANSIT : Transition matrix compilation"
PRINT STRING$(80, "=")
OPEN "tri-inc.dat" FOR INPUT AS #1
'initialize diameter class bounds, declare arrays, set transition period (years)
ndc% = 8
DIM dc(ndc%), tm(ndc%, ndc%), ntrees(ndc%)
DATA 5,10,20,40,60,80,100,120
FOR i% = 1 TO ndc%
  READ dc(i%)
NEXT i%
tp = 5
'loop through data counting trees by transitions and initial class totals
DO
  LOCATE 4, 1: PRINT nrec%; : nrec% = nrec% + 1
  INPUT #1, diam, dinc, dead$
  IF EOF(1) THEN EXIT DO
  ' get initial class, count trees in class
  f% = dclass%(diam)
  ntrees(f%) = ntrees(f%) + 1
  ' for live trees, get final class, count transition from class i to j
  IF dead$ = "" THEN
    'estimate final size from increment and transition period
    diam2 = diam + dinc * tp
    t% = dclass%(diam2)
    tm(t%, f%) = tm(t%, f%) + 1
  END IF
LOOP
'adjust transition tallies to proportions and write to output file
OPEN "transit.prn" FOR OUTPUT AS #2
FOR t% = 1 TO ndc%
  FOR f% = 1 TO ndc%
    IF ntrees(f%) > 0 THEN
      tm(t%, f%) = tm(t%, f%) / ntrees(f%)
    ELSE
      tm(t%, f%) = 0
    END IF
    PRINT #2, USING "#.####"; tm(t%, f%);
    IF f% < ndc% THEN PRINT #2, ",";  ELSE PRINT #2,
  NEXT f%
NEXT t%
END

FUNCTION dclass% (diam)
'get index for a diameter class using bounds in array dc()
SHARED dc(), ndc%
IF diam < dc(1) THEN dclass% = 0: EXIT FUNCTION
FDR k% = 1 TO ndc% - 1
  IF diam >= dc(k%) AND diam < dc(k% + 1) THEN dclass% = k%: EXIT FUNCTION
NEXT k%
dclass% = ndc%
END FUNCTION
```

Example 35 *TRANSIT program to compile transition matrices from tree increment data* of this type directly from diameter increment data. The program expects as inputs tree data for a given species or species group. Each line of the input data file should have initial diameter, annual diameter increment, and a mortality code. These data can be generated from an XBASE file such as the one in Example 19 on page 43. The initial diameter (DPOM), diameter increment (DINC) and mortality code (DEAD) are extracted as for the species *Triplochiton scleroxylon* to a text file called TRI-INC.DAT using the following XBASE commands:

```
USE GHPSPINC
SET ORDER TO SPP
FIND TRI
COPY TO TRI-INC.DAT TYPE DELIM FIELDS DPOM,DINC,DEAD WHILE SPP="TRI"
```

Line 5 of the program listing shows how this text file, called TRI-INC.DAT, is opened by the BASIC program.

The diameter classes that correspond to the rows and columns of the transition matrix are defined in the DATA statement on line 9. The method used in this program allows variable intervals between classes. This can be contrasted with the method shown on page 42, which requires fixed-width diameter classes. The function DCLASS% on lines 44-52 of the program checks which class a given diameter is in, using the values in the array DC read from the DATA statement at lines 10-12.

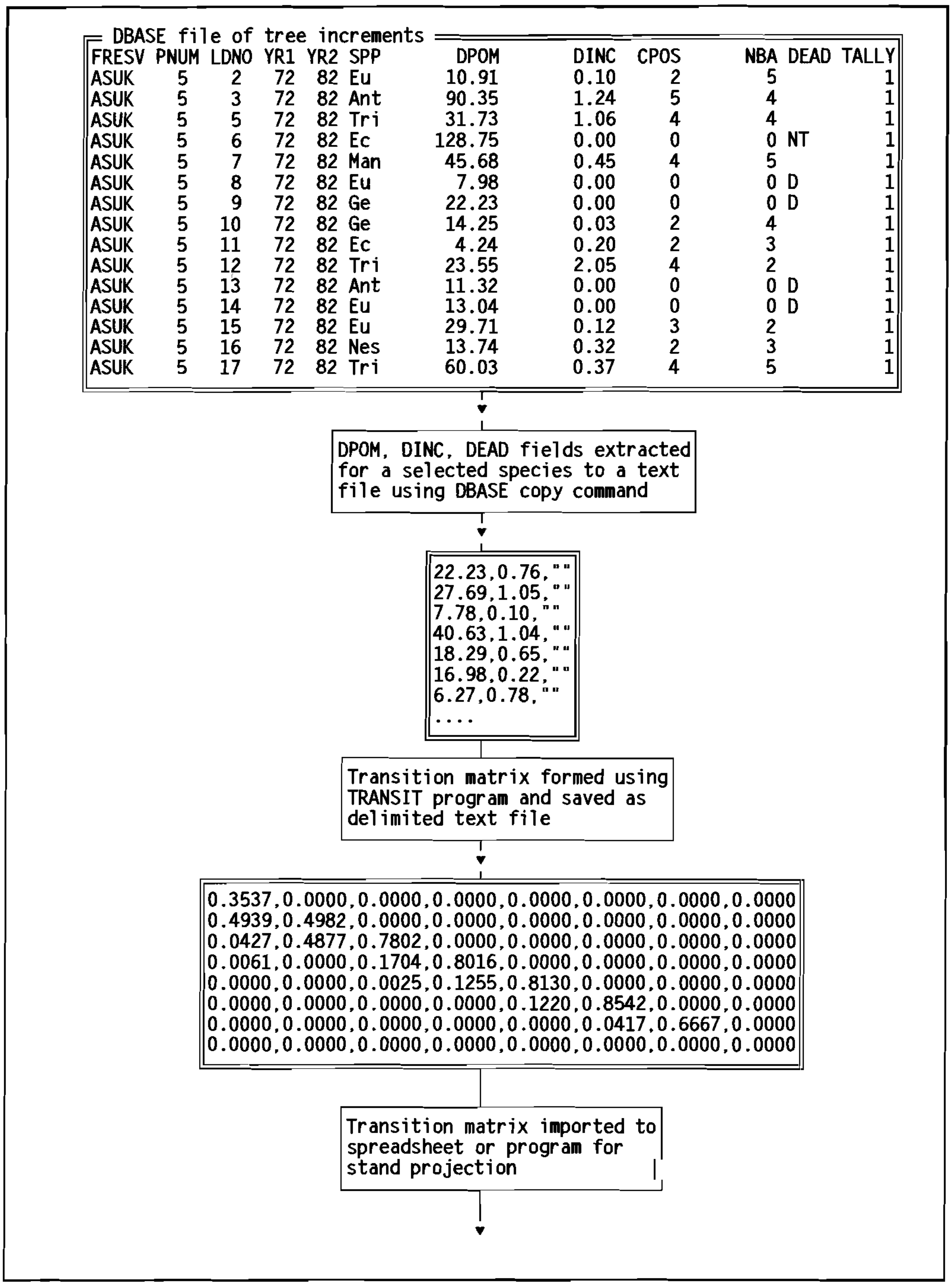
Line 13 of the program defines the transition period (TP) as 5 years. Since the method involves linear extrapolation of annual diameter increments that are themselves derived as periodic averages over intervals of 5-10 years, this is reasonably accurate. However, the TP period should not be set to a value much greater than the actual period of measurement used on the PSPs.

Lines 15-29 loop through the input data, line by line, and count the trees in each transition from diameter class F% to class T%. Dead trees are not counted in the transition matrix TM, but are counted in the NTREES array which is used at the end to adjust the raw tallies.

Lines 31-42 output the computed transition matrix, correcting the raw tallies for the total number of trees alive in each class at the start of the period. The output format is designed to be read directly by either another BASIC program, or a spreadsheet.

The data transformations involved, from the original PSP data to the final transition matrix, are summarized in Example 36. This also shows a section of the input file, to illustrate the format used, and the complete transition matrix as it is output by the program.

To demonstrate how the transition matrix can be used to construct a simple stand model, it has been imported into a Lotus 1-2-3 spreadsheet, as shown in Example 37. The transition matrix will have been saved by the TRANSIT program in a file called TRANSIT.PRN (see line 31 of the program). This can be imported into



```
DBASE file of tree increments
FRESV PNUM LDNO YR1 YR2 SPP        DPOM        DINC  CPOS       NBA DEAD TALLY
ASUK    5    2   72  82 Eu        10.91        0.10    2         5           1
ASUK    5    3   72  82 Ant       90.35        1.24    5         4           1
ASUK    5    5   72  82 Tri       31.73        1.06    4         4           1
ASUK    5    6   72  82 Ec       128.75        0.00    0         0 NT        1
ASUK    5    7   72  82 Man       45.68        0.45    4         5           1
ASUK    5    8   72  82 Eu         7.98        0.00    0         0 D         1
ASUK    5    9   72  82 Ge        22.23        0.00    0         0 D         1
ASUK    5   10   72  82 Ge        14.25        0.03    2         4           1
ASUK    5   11   72  82 Ec         4.24        0.20    2         3           1
ASUK    5   12   72  82 Tri       23.55        2.05    4         2           1
ASUK    5   13   72  82 Ant       11.32        0.00    0         0 D         1
ASUK    5   14   72  82 Eu        13.04        0.00    0         0 D         1
ASUK    5   15   72  82 Eu        29.71        0.12    3         2           1
ASUK    5   16   72  82 Nes       13.74        0.32    2         3           1
ASUK    5   17   72  82 Tri       60.03        0.37    4         5           1
```

DPOM, DINC, DEAD fields extracted for a selected species to a text file using DBASE copy command

```
22.23,0.76,""
27.69,1.05,""
7.78,0.10,""
40.63,1.04,""
18.29,0.65,""
16.98,0.22,""
6.27,0.78,""
....
```

Transition matrix formed using TRANSIT program and saved as delimited text file

```
0.3537,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
0.4939,0.4982,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
0.0427,0.4877,0.7802,0.0000,0.0000,0.0000,0.0000,0.0000
0.0061,0.0000,0.1704,0.8016,0.0000,0.0000,0.0000,0.0000
0.0000,0.0000,0.0025,0.1255,0.8130,0.0000,0.0000,0.0000
0.0000,0.0000,0.0000,0.0000,0.1220,0.8542,0.0000,0.0000
0.0000,0.0000,0.0000,0.0000,0.0000,0.0417,0.6667,0.0000
0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
```

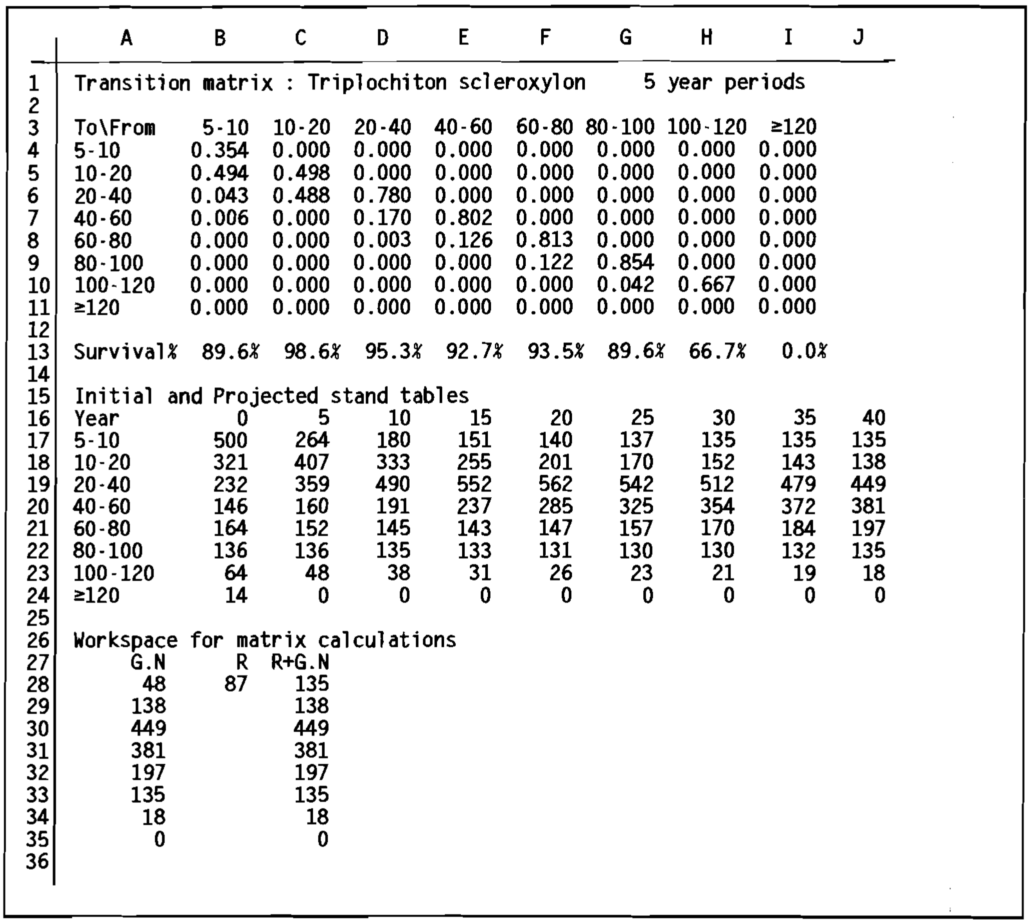
Transition matrix imported to spreadsheet or program for stand projection

Example 36 *Data transformations from tree increment data to a transition matrix*

a Lotus spreadsheet* using the /File Import Number menu sequence. The import will occur at the current cell position, over-writing any existing data. For the example, the cursor should therefore be positioned at cell B4.

The stand table to be projected is then entered in columnar form. This is shown

* This discussion relates to Lotus 1-2-3 version 2.2. Later versions may use different menu sequences.



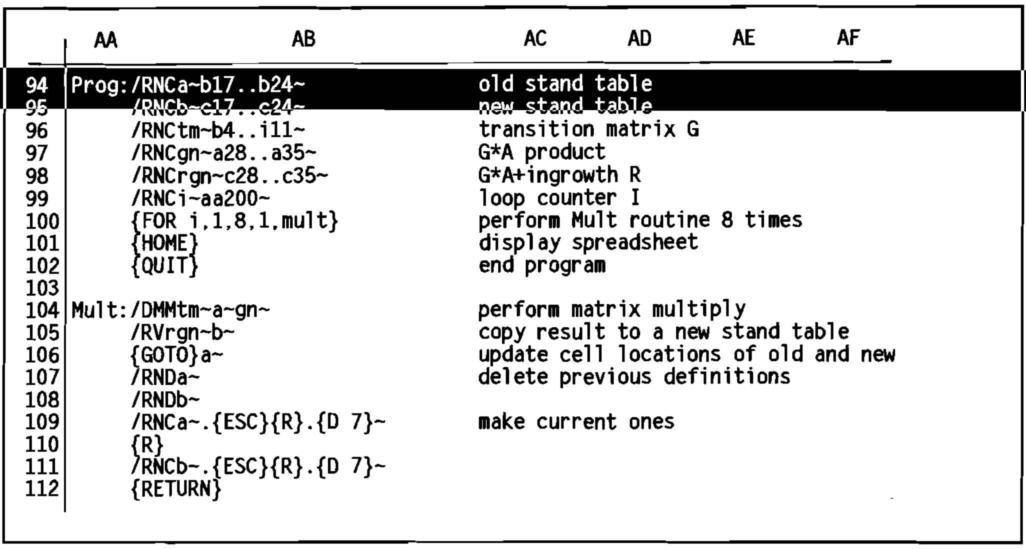
```
     A           B      C      D      E      F      G      H      I      J
 1   Transition matrix : Triplochiton scleroxylon     5 year periods
 2
 3   To\From     5-10   10-20  20-40  40-60  60-80  80-100 100-120  ≥120
 4   5-10        0.354  0.000  0.000  0.000  0.000  0.000  0.000  0.000
 5   10-20       0.494  0.498  0.000  0.000  0.000  0.000  0.000  0.000
 6   20-40       0.043  0.488  0.780  0.000  0.000  0.000  0.000  0.000
 7   40-60       0.006  0.000  0.170  0.802  0.000  0.000  0.000  0.000
 8   60-80       0.000  0.000  0.003  0.126  0.813  0.000  0.000  0.000
 9   80-100      0.000  0.000  0.000  0.000  0.122  0.854  0.000  0.000
10   100-120     0.000  0.000  0.000  0.000  0.000  0.042  0.667  0.000
11   ≥120        0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
12
13   Survival%   89.6%  98.6%  95.3%  92.7%  93.5%  89.6%  66.7%   0.0%
14
15   Initial and Projected stand tables
16   Year            0      5     10     15     20     25     30     35     40
17   5-10          500    264    180    151    140    137    135    135    135
18   10-20         321    407    333    255    201    170    152    143    138
19   20-40         232    359    490    552    562    542    512    479    449
20   40-60         146    160    191    237    285    325    354    372    381
21   60-80         164    152    145    143    147    157    170    184    197
22   80-100        136    136    135    133    131    130    130    132    135
23   100-120        64     48     38     31     26     23     21     19     18
24   ≥120           14      0      0      0      0      0      0      0      0
25
26   Workspace for matrix calculations
27         G.N       R  R+G.N
28          48      87    135
29         138           138
30         449           449
31         381           381
32         197           197
33         135           135
34          18            18
35           0             0
36
```

Example 37 *Spreadsheet example of transition matrix model*

in the example in cells B17:B24. The matrix product **G.N** (see equation {35}) is formed using the /Data Matrix Multiply menu selection, defining the prefactor (**G**) as cells B4:I11, and the postfactor (**N**) as cells B17:B24, that is, the stand table that has been entered. The result is placed in cells A28:A35. After the matrix multiplication, this will give the projected stand table without accounting for recruitment or harvesting. It should be noted that mortality is however implicit in the stand table. Cells B13:I13 have been defined to contain the sums of each row of the transition matrix (as in equation {36}) using the formula @SUM(x17..x24) where x is the column letter in the spreadsheet. These show net survival of trees originating in each diameter class at the end of a growth period.

Recruitment is assumed in this model to be constant in each time period, and is entered as the vector **R** in cells B28:B35. It can be seen that a constant value of 87 trees/km^2 is assumed to grow into the first class. This and the original stand table data are the same as those used in Example 33 on page 78, to provide a direct comparison of the results of the two modelling methods.

The result of the vector addition (**R + G.N**) is given in cells C28:C35. It is defined by using the spreadsheet formula Ax+Bx for each cell Cx, where x is the row number 28:35. This is the projected stand table, including ingrowth, for the first time period. It is copied back into the main table at cells C17:C24. The copy is accomplished in Lotus 1-2-3 using the /Range Value command, rather than /Copy.



```
     AA            AB                    AC         AD         AE         AF
94   Prog:/RNCa~b17..b24~                old stand table
95        /RNCb~c17..c24~                new stand table
96        /RNCtm~b4..i11~                transition matrix G
97        /RNCgn~a28..a35~               G*A product
98        /RNCrgn~c28..c35~              G*A+ingrowth R
99        /RNCi~aa200~                   loop counter I
100       {FOR i,1,8,1,mult}             perform Mult routine 8 times
101       {HOME}                         display spreadsheet
102       {QUIT}                         end program
103
104  Mult:/DMMtm~a~gn~                   perform matrix multiply
105       /RVrgn~b~                      copy result to a new stand table
106       {GOTO}a~                       update cell locations of old and new
107       /RNDa~                         delete previous definitions
108       /RNDb~
109       /RNCa~.{ESC}{R}.{D 7}~         make current ones
110       {R}
111       /RNCb~.{ESC}{R}.{D 7}~
112       {RETURN}
```

Example 38 *Lotus 1-2-3 macro to generate transition matrix projections*

The former copies only the numerical values in cells C28:C35; the latter would copy and adjust the formulae, giving erroneous results. The process is repeated for each projection year, redefining the input vector to the /Data Matrix Multiply command and the output area of the /Range Value copy.

These operations are tedious and repetitive for each projection year. The Lotus 1-2-3 Macro shown in Example 38 automates them. It is written as shown in cells whose row and column numbers are outside the main spreadsheet, so that any insertions or deletions do not affect the macro. It defines the matrices and vectors used by name with the /Range Name Create command in cells AB94:AB99. The subroutine Mult is then repeated 8 times (for projection years 5-40) with the {FOR} command in cell AB100.

The *Mult* subroutine performs the /Data Matrix Multiply and /Range Value commands as discussed above for the initial projection period, and then redefines the applicable cell ranges. The statements in cells AB106:AB111 redefine the ranges using the /Range Name Delete and /Range Name Create commands.

In order to use this macro, after creating the spreadsheet and entering the program, two ranges must be named. *Prog* should be defined as cells AB94:AB102; and *Mult* as cells AB104:AB112. The Alt-F3 (Run macro) key can then be used to execute *Prog*. To observe how the macro works, the Alt-F2 key can be used to place the spreadsheet in single step program mode.

By modifying values for the initial stand table, the transition matrix or the regeneration vector, and then re-running *Prog*, a revised set of projections will be obtained.

This procedure, or an appropriate variation of it, allows a transition matrix model to be implemented using a spreadsheet in a manner that is analogous to the simple stand table projection speadsheet in Example 33. Both examples use the same initial stand table, recruitment assumption, and PSP data to derive the growth estimates. However, it will be noted that the results obtained are somewhat different. The variation may be attributed principally to the following factors:

- The transition matrix compiled directly from data does not have any trees in the 120 cm class, and it is therefore impossible for trees to reach this size. They will attain the 100 cm class, and then die there.

- The UDMI assumptions (see page 77) are over-optimistic in their assessment of growth. For example, in the matrix model, net outgrowth from the the first class is

  $$(0.494+0.043+0.006) = 54.3\%$$

  whilst outgrowth under the UDMI assumptions is 68.6%.

The problem of empty cells in a transition matrix due to lack of data can be solved in two ways:

- The classes can be re-defined to ensure that all classes include some trees from the PSP data. In the present example, the 120cm+ class should be eliminated, with the largest class represented being 100cm+.

- Growth functions can be fitted to the data, with the transition matrix being built indirectly from them. This smoothes over deficiencies in the data, but also introduces assumptions into the model that may bias it.

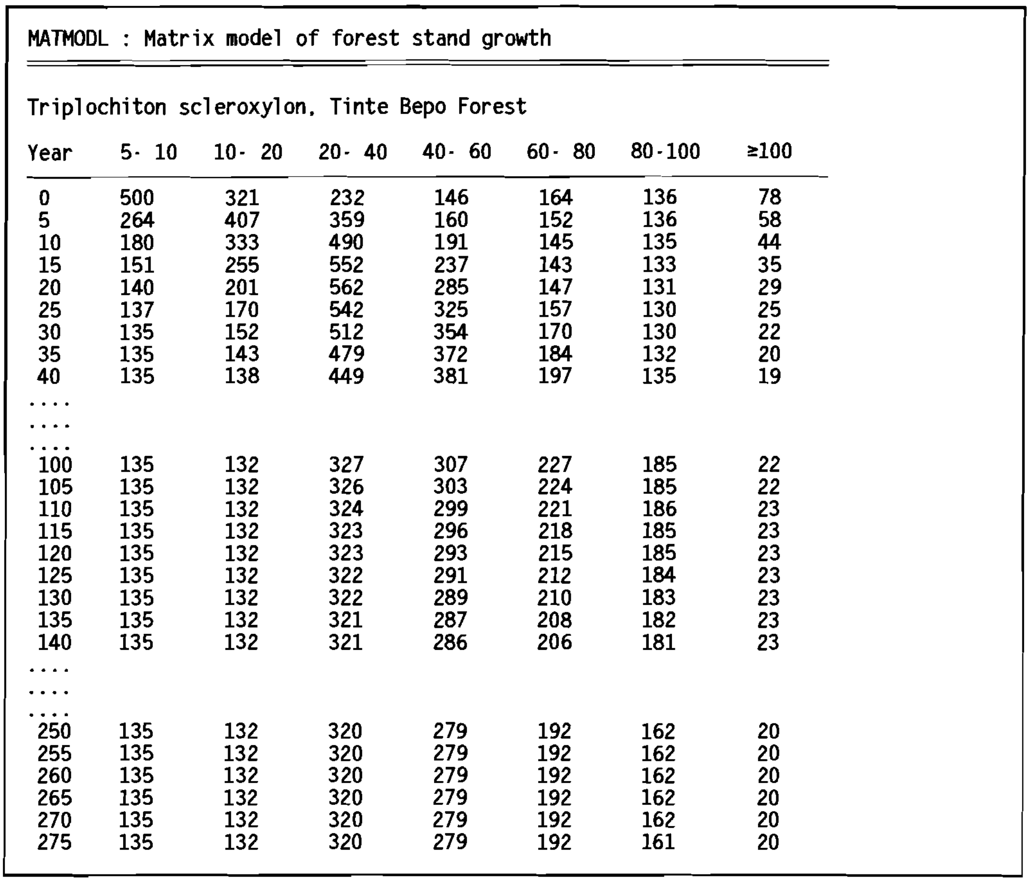
Description	Data file contents
Run title	"Triplochiton scleroxylon, Tinte Bepo Forest"
No. diam classes	7
Diam class bounds	5,10,20,40,60,80,100
Transition matrix	0.3537,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
	0.4939,0.4982,0.0000,0.0000,0.0000,0.0000,0.0000
	0.0427,0.4877,0.7802,0.0000,0.0000,0.0000,0.0000
	0.0061,0.0000,0.1704,0.8016,0.0000,0.0000,0.0000
	0.0000,0.0000,0.0025,0.1255,0.8130,0.0000,0.0000
	0.0000,0.0000,0.0000,0.0000,0.1220,0.8542,0.0000
	0.0000,0.0000,0.0000,0.0000,0.0000,0.0417,0.6667
Regeneration vector	87,0,0,0,0,0,0
Initial stand table	500,321,232,146,164,136,78

Example 39 *Input data file for MATMODL program*

The spreadsheet representation of a matrix model is obviously cumbersome compared with an equivalent computer program. Example 40 shows a BASIC program called MATMODL which performs the necessary calculations. The data file used to run this program is shown in Example 39. A title, the number of diameter classes, and the class lower bounds are defined in the first three lines. The transition matrix is edited-in using the output file TRANSIT.PRN from the TRANSIT program. A vector giving recruitment per time period follows, and then the initial stand table.

```
'Matrix model for forest stand growth
CLS
PRINT "MATMODL : Matrix model of forest stand growth"
PRINT STRING$(80, "="): PRINT
'read parameter file - title, no. classes, class bounds, transition matrix
'and initial stand table
OPEN "matmodl.dat" FOR INPUT AS #1
INPUT #1, title$: PRINT title$
INPUT #1, ndc%
DIM dc(ndc%), tm(ndc%, ndc%), reg(ndc%), sta(ndc%), stb(ndc%)
FOR i% = 1 TO ndc%: INPUT #1, dc(i%): NEXT i%
FOR i% = 1 TO ndc%
  FOR j% = 1 TO ndc%: INPUT #1, tm(i%, j%): NEXT j%
NEXT i%
FOR i% = 1 TO ndc%: INPUT #1, reg(i%): NEXT i%
FOR i% = 1 TO ndc%: INPUT #1, sta(i%): NEXT i%
'write diameter class headings
PRINT
PRINT "Year  ";
FOR i% = 1 TO ndc%
  IF i% < ndc% THEN
    PRINT USING "###_-###  "; dc(i%); dc(i% + 1);
  ELSE
    PRINT USING "  ≥###"; dc(i%)
  END IF
NEXT i%
PRINT STRING$(6 + 9 * ndc%, "—")
year = 0
DO
  'print current table
  PRINT year; TAB(6);
  FOR i% = 1 TO ndc%: PRINT USING "  ####   "; sta(i%); : NEXT i%
  PRINT
  'matrix operations to project stand
  CALL mmult(tm(), sta(), stb())
  CALL madd(stb(), reg(), sta())
  'press enter to simulate next year, anything else to stop
  year = year + 5
LOOP UNTIL INPUT$(1) <> CHR$(13)
END

SUB madd (a(), b(), c())
'adds vector a to vector b giving vector c
n% = UBOUND(a, 1)
FOR i% = 1 TO n%
  c(i%) = a(i%) + b(i%)
NEXT i%
END SUB

SUB mmult (a(), b(), c())
'multiplies square matrix a and vector b into vector c
n% = UBOUND(a, 1)
FOR i% = 1 TO n%: c(i%) = 0: NEXT i%
FOR i% = 1 TO n%
  FOR j% = 1 TO n%
    c(j%) = c(j%) + a(j%, i%) * b(i%)
  NEXT j%
NEXT i%
END SUB
```

Example 40 *BASIC program for a transition matrix model of forest growth*



```
MATMODL : Matrix model of forest stand growth
===================================================================================

Triplochiton scleroxylon, Tinte Bepo Forest

Year    5- 10    10- 20    20- 40    40- 60    60- 80    80-100     ≥100
-----------------------------------------------------------------------------------
0       500      321       232       146       164       136        78
5       264      407       359       160       152       136        58
10      180      333       490       191       145       135        44
15      151      255       552       237       143       133        35
20      140      201       562       285       147       131        29
25      137      170       542       325       157       130        25
30      135      152       512       354       170       130        22
35      135      143       479       372       184       132        20
40      135      138       449       381       197       135        19
....
....
....
100     135      132       327       307       227       185        22
105     135      132       326       303       224       185        22
110     135      132       324       299       221       186        23
115     135      132       323       296       218       185        23
120     135      132       323       293       215       185        23
125     135      132       322       291       212       184        23
130     135      132       322       289       210       183        23
135     135      132       321       287       208       182        23
140     135      132       321       286       206       181        23
....
....
....
250     135      132       320       279       192       162        20
255     135      132       320       279       192       162        20
260     135      132       320       279       192       162        20
265     135      132       320       279       192       162        20
270     135      132       320       279       192       162        20
275     135      132       320       279       192       161        20
```

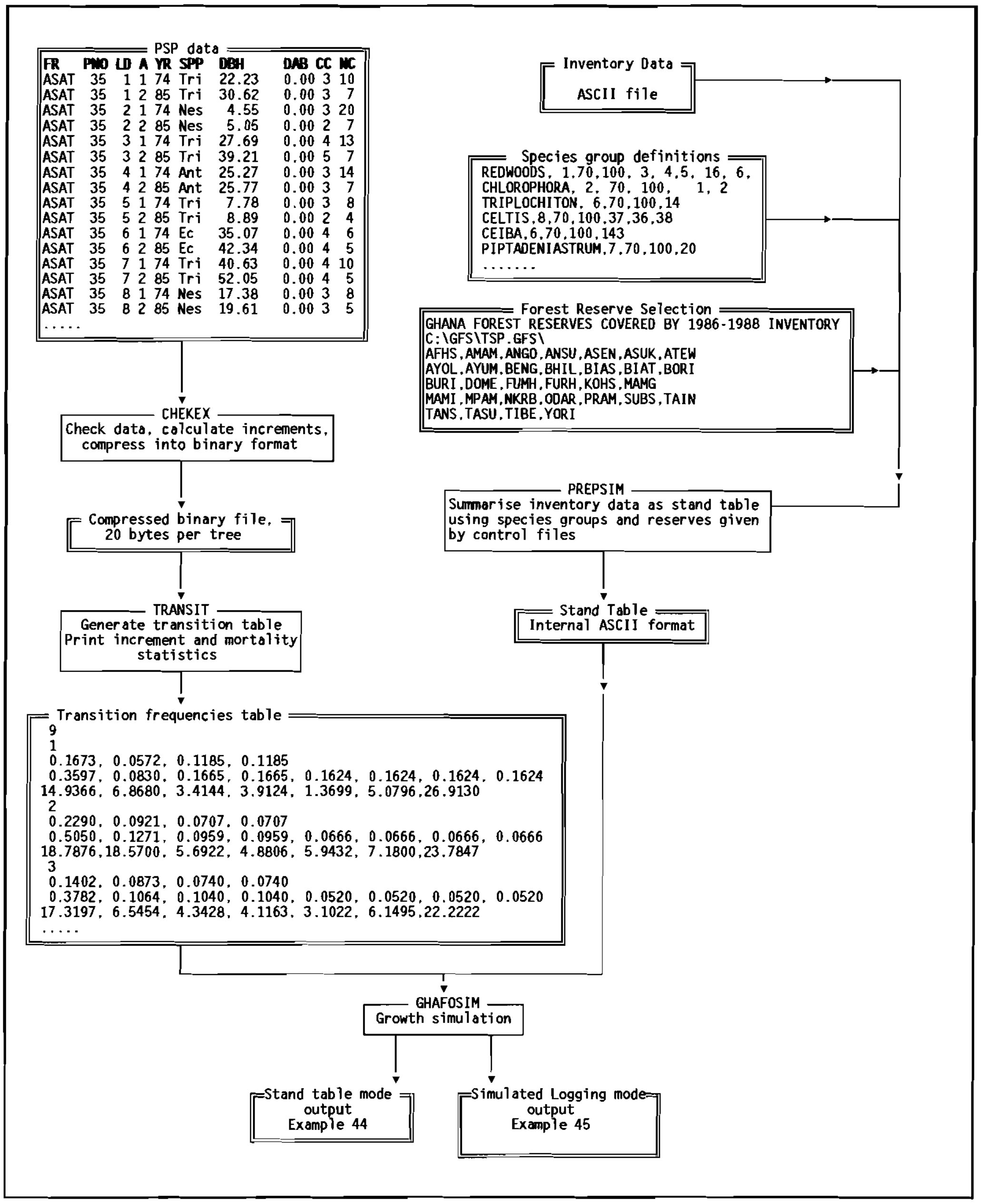
Example 41 *Projected stand tables from the transition matrix model, tending towards an equilbrium distribution after 250 years*

The output from the program is shown in Example 41. This has been edited, showing "...." where lines have been removed to abbreviate the output. It can be seen that the diameter class distribution tends towards a stable state, which is reached after about 250 simulated years (50 projection periods). Experiments with the input parameters will show that the shape of the equilibrium distribution depends on the transition matrix, whilst its magnitude depends on the level of recruitment. The equilibrium distribution is independent of the initial stand table.

This type of matrix model is designed for a single species only. Both MATMODL and TRANSIT can be modified to operate with many species, building separate transition matrices for each and running them to produce independent stand tables. However, it will be noted there is no interaction between growth rate and stand density in this type of model.

3.2.3 *GHAFOSIM : A case study*

GHAFOSIM is a model developed to assist natural forest management in Ghana. It typifies the features that may be expected in a diameter class projection model. It is also a model that represents a compromise between the 'state of the art' in terms of modelling technique, and what is actually possible due to limitations of available data and computer systems. A complete specification is given in Alder (1990).


PSP data
CHEKEX
Check data, calculate increments, compress into binary format
Compressed binary file, 20 bytes per tree
TRANSIT
Generate transition table Print increment and mortality statistics
Transition frequencies table
Inventory Data
ASCII file
Species group definitions
Forest Reserve Selection
PREPSIM
Summarise inventory data as stand table using species groups and reserves given by control files
Stand Table
Internal ASCII format
GHAFOSIM
Growth simulation
Stand table mode output Example 44
Simulated Logging mode output Example 45

```
FR   PNO LD A YR SPP  DBH     DAB CC NC
ASAT  35  1 1 74 Tri  22.23   0.00 3 10
ASAT  35  1 2 85 Tri  30.62   0.00 3  7
ASAT  35  2 1 74 Nes   4.55   0.00 3 20
ASAT  35  2 2 85 Nes   5.05   0.00 2  7
ASAT  35  3 1 74 Tri  27.69   0.00 4 13
ASAT  35  3 2 85 Tri  39.21   0.00 5  7
ASAT  35  4 1 74 Ant  25.27   0.00 3 14
ASAT  35  4 2 85 Ant  25.77   0.00 3  7
ASAT  35  5 1 74 Tri   7.78   0.00 3  8
ASAT  35  5 2 85 Tri   8.89   0.00 2  4
ASAT  35  6 1 74 Ec   35.07   0.00 4  6
ASAT  35  6 2 85 Ec   42.34   0.00 4  5
ASAT  35  7 1 74 Tri  40.63   0.00 4 10
ASAT  35  7 2 85 Tri  52.05   0.00 4  5
ASAT  35  8 1 74 Nes  17.38   0.00 3  8
ASAT  35  8 2 85 Nes  19.61   0.00 3  5
.....
```

```
REDWOODS, 1,70,100, 3, 4,5, 16, 6,
CHLOROPHORA, 2, 70, 100,   1, 2
TRIPLOCHITON, 6,70,100,14
CELTIS,8,70,100,37,36,38
CEIBA,6,70,100,143
PIPTADENIASTRUM,7,70,100,20
.......
```

```
GHANA FOREST RESERVES COVERED BY 1986-1988 INVENTORY
C:\GFS\TSP.GFS\
AFHS,AMAM,ANGO,ANSU,ASEN,ASUK,ATEW
AYOL,AYUM,BENG,BHIL,BIAS,BIAT,BORI
BURI,DOME,FUMH,FURH,KOHS,MAMG
MAMI,MPAM,NKRB,ODAR,PRAM,SUBS,TAIN
TANS,TASU,TIBE,YORI
```

```
 9
 1
 0.1673, 0.0572, 0.1185, 0.1185
 0.3597, 0.0830, 0.1665, 0.1665, 0.1624, 0.1624, 0.1624, 0.1624
14.9366, 6.8680, 3.4144, 3.9124, 1.3699, 5.0796,26.9130
 2
 0.2290, 0.0921, 0.0707, 0.0707
 0.5050, 0.1271, 0.0959, 0.0959, 0.0666, 0.0666, 0.0666, 0.0666
18.7876,18.5700, 5.6922, 4.8806, 5.9432, 7.1800,23.7847
 3
 0.1402, 0.0873, 0.0740, 0.0740
 0.3782, 0.1064, 0.1040, 0.1040, 0.0520, 0.0520, 0.0520, 0.0520
17.3197, 6.5454, 4.3428, 4.1163, 3.1022, 6.1495,22.2222
 .....
```

Example 42 ***Data files and modules forming part of the GHAFOSIM system***

The model used a series of PSPs established in 1968, and remeasured over various periods until 1983 (Baidoe, 1968). These plots were designed as tree increment plots, measuring 50 'Leading Desirable' trees (LDs) on each plot. The LDs were selected on criteria of species, from a set of 24 commercial species (Table 3), and their potential as 'final crop trees'. Consequences of this methodology were that:

- No information was available on many important species.
- No information was available on whole stand growth. Only individual tree increment information was recorded.
- The selection of sample trees was subjective, making assessment of

recruitment rates impossible to determine.

The computer resources available at the time of program development (1989-90) constituted a further constraint on its design. It was originally limited by the need to operate effectively on an IBM XT-class machine without hard disk. All the programs were written in Turbo BASIC and designed to run as compiled .EXE commands. The various steering data files used could be created by any standard text editor.

Trade name	Botanical name	Species codes	
		PSPs	TSPs
FD Class I			
Odum	Milicia (Chlorophora) spp.	CHL	1,2
Edinam	Entandophragma angolense	EA	3
Sapele	Entandophragma cyclindricum	EC	4
Utile	Entandophragma utile	EU	5
Mohogany	Khaya ivorensis	KI	8
Mahogany	Khaya anthotheca	KA	6
Makore	Tieghemella heckelii	TIE	9
Kusia	Nauclea diderichii	NAU	10
Afromosia	Pericopsis elata	PER	11
Walnnt	Lovoa trichilioides	LOV	12
Emire	Terminalia ivorensis	TI	13
Wawa	Triplochiton scleroxylon	TRI	14
Nyankom	Heretiera utilis	TAR	15
FD Class II			
Candollei	Entandophragma candollei	ECL	16
Guarea	Guarea cedrata	GC	17
Guarea	Guarea thompsonii	GT	18
Kaku	Lophira elata	LOP	19
Dahoma	Piptadeniastrum africanum	P	20
Kyenkyen	Antiaris toxicaria	ANT	21
Hyedua	Guibourtia ehie	GE	22
Mansonia	Mansonia altissima	MAN	23
Subaha	Mitragyna stipulosa	MIS	24
Danta	Nesogordonia Papaverifera	NES	26
Avodire	Turraeanthus africanus	TUR	27

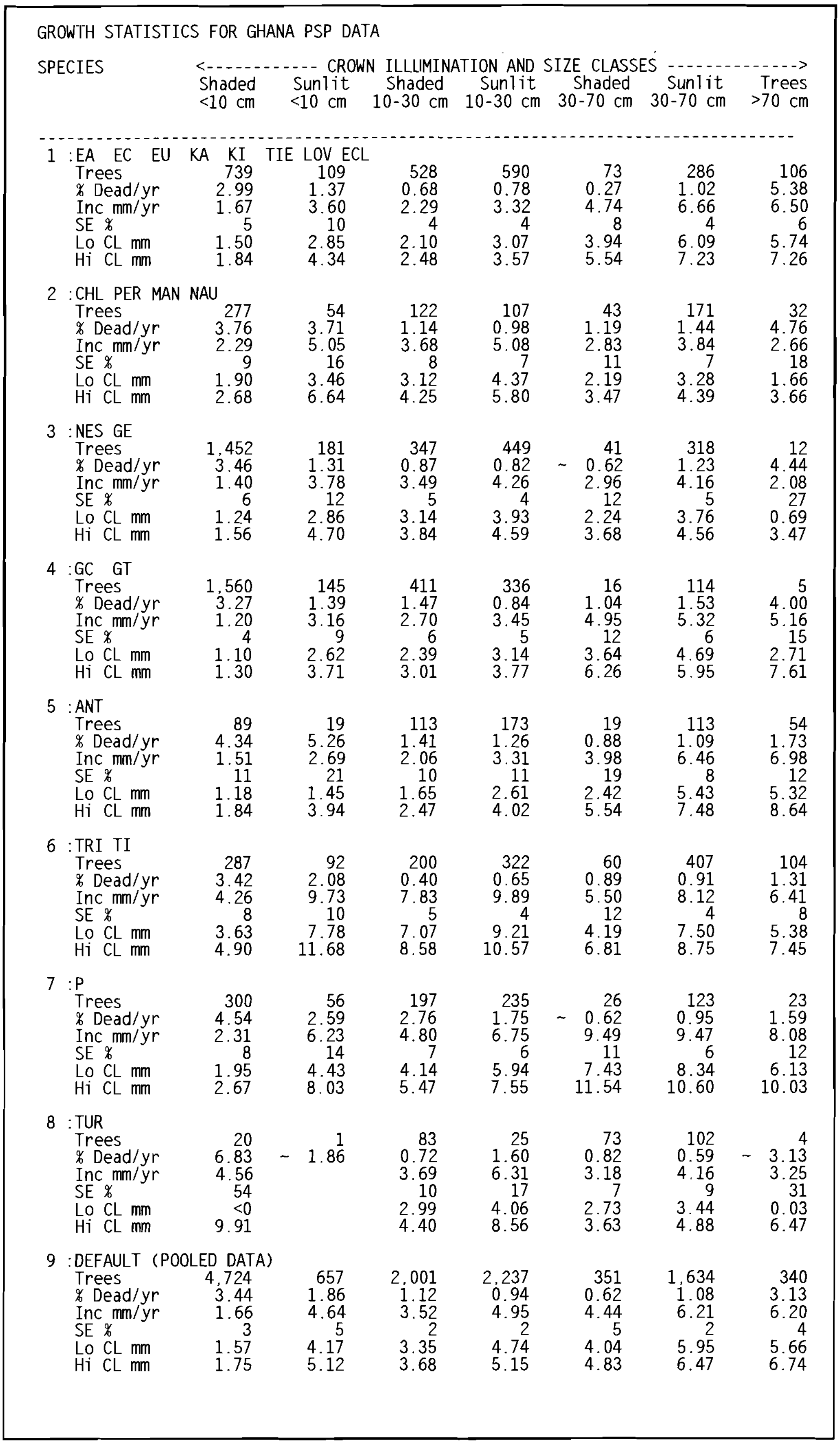
Table 3 *Species codes in GHAFOSIM examples*

The growth model subroutine that is the core of GHAFOSIM performs stand table projection for aggregate forest data, compiled from forest inventory plots using transition frequencies and mortality rates to determine net outgrowth from each class. The stand table distinguishes between trees which are sunlit and those which are mostly or completely shaded, and uses different growth and mortality rates for these categories. Trees can be demoted from sunlit to shade as a result of excessive stocking, or promoted from shade to sun following logging. The maximum numbers of sunlit trees found in each size class were determined from an analysis of temporary plot data. In future revisions of the program, this feature will be refined by incorporating stem:crown diameter ratios determined by Foli (1993), and limiting sunlit trees to those which give a crown projection area of 100%.

The growth rates are presented as average increments for documentation (see Example 43), but are compiled internally as transition frequencies, thus avoiding the bias associated with the UDMI assumptions (see page 77). Species are grouped during analysis to give a limited number of growth models. Each model is assigned a serial number. An internal table of the transition frequencies and mortalities for each size and shade class is maintained for each model.

The GHAFOSIM system comprises 4 independent program modules, as shown in Example 42. This organization separates tasks that involve the time consuming and infrequent compilation of PSP or TSP data from those required for each simulation run of the model. The modules are as follows:

CHEKEX This checks the original PSP data and calculates increments. The PSP data is stored as shown, as ASCII fixed-length records, sorted by forest, plot, tree, and measurement year. Only the tree DBH and crown class (CC) are used in the analysis. The column NC (number of competitors) is a zone count score which unfortunately has proved to have no



```
GROWTH STATISTICS FOR GHANA PSP DATA

SPECIES        <------------ CROWN ILLUMINATION AND SIZE CLASSES ------------->
                Shaded   Sunlit    Shaded    Sunlit    Shaded    Sunlit    Trees
                <10 cm   <10 cm   10-30 cm  10-30 cm  30-70 cm  30-70 cm  >70 cm

----------------------------------------------------------------------------
 1 :EA  EC  EU  KA  KI  TIE LOV ECL
    Trees         739      109       528       590        73       286       106
    % Dead/yr    2.99     1.37      0.68      0.78      0.27      1.02      5.38
    Inc mm/yr    1.67     3.60      2.29      3.32      4.74      6.66      6.50
    SE %            5       10         4         4         8         4         6
    Lo CL mm     1.50     2.85      2.10      3.07      3.94      6.09      5.74
    Hi CL mm     1.84     4.34      2.48      3.57      5.54      7.23      7.26

 2 :CHL PER MAN NAU
    Trees         277       54       122       107        43       171        32
    % Dead/yr    3.76     3.71      1.14      0.98      1.19      1.44      4.76
    Inc mm/yr    2.29     5.05      3.68      5.08      2.83      3.84      2.66
    SE %            9       16         8         7        11         7        18
    Lo CL mm     1.90     3.46      3.12      4.37      2.19      3.28      1.66
    Hi CL mm     2.68     6.64      4.25      5.80      3.47      4.39      3.66

 3 :NES GE
    Trees       1,452      181       347       449        41       318        12
    % Dead/yr    3.46     1.31      0.87      0.82    ~ 0.62      1.23      4.44
    Inc mm/yr    1.40     3.78      3.49      4.26      2.96      4.16      2.08
    SE %            6       12         5         4        12         5        27
    Lo CL mm     1.24     2.86      3.14      3.93      2.24      3.76      0.69
    Hi CL mm     1.56     4.70      3.84      4.59      3.68      4.56      3.47

 4 :GC  GT
    Trees       1,560      145       411       336        16       114         5
    % Dead/yr    3.27     1.39      1.47      0.84      1.04      1.53      4.00
    Inc mm/yr    1.20     3.16      2.70      3.45      4.95      5.32      5.16
    SE %            4        9         6         5        12         6        15
    Lo CL mm     1.10     2.62      2.39      3.14      3.64      4.69      2.71
    Hi CL mm     1.30     3.71      3.01      3.77      6.26      5.95      7.61

 5 :ANT
    Trees          89       19       113       173        19       113        54
    % Dead/yr    4.34     5.26      1.41      1.26      0.88      1.09      1.73
    Inc mm/yr    1.51     2.69      2.06      3.31      3.98      6.46      6.98
    SE %           11       21        10        11        19         8        12
    Lo CL mm     1.18     1.45      1.65      2.61      2.42      5.43      5.32
    Hi CL mm     1.84     3.94      2.47      4.02      5.54      7.48      8.64

 6 :TRI TI
    Trees         287       92       200       322        60       407       104
    % Dead/yr    3.42     2.08      0.40      0.65      0.89      0.91      1.31
    Inc mm/yr    4.26     9.73      7.83      9.89      5.50      8.12      6.41
    SE %            8       10         5         4        12         4         8
    Lo CL mm     3.63     7.78      7.07      9.21      4.19      7.50      5.38
    Hi CL mm     4.90    11.68      8.58     10.57      6.81      8.75      7.45

 7 :P
    Trees         300       56       197       235        26       123        23
    % Dead/yr    4.54     2.59      2.76      1.75    ~ 0.62      0.95      1.59
    Inc mm/yr    2.31     6.23      4.80      6.75      9.49      9.47      8.08
    SE %            8       14         7         6        11         6        12
    Lo CL mm     1.95     4.43      4.14      5.94      7.43      8.34      6.13
    Hi CL mm     2.67     8.03      5.47      7.55     11.54     10.60     10.03

 8 :TUR
    Trees          20        1        83        25        73       102         4
    % Dead/yr    6.83   ~ 1.86      0.72      1.60      0.82      0.59    ~ 3.13
    Inc mm/yr    4.56               3.69      6.31      3.18      4.16      3.25
    SE %           54                 10        17         7         9        31
    Lo CL mm       <0               2.99      4.06      2.73      3.44      0.03
    Hi CL mm     9.91               4.40      8.56      3.63      4.88      6.47

 9 :DEFAULT (POOLED DATA)
    Trees       4,724      657     2,001     2,237       351     1,634       340
    % Dead/yr    3.44     1.86      1.12      0.94      0.62      1.08      3.13
    Inc mm/yr    1.66     4.64      3.52      4.95      4.44      6.21      6.20
    SE %            3        5         2         2         5         2         4
    Lo CL mm     1.57     4.17      3.35      4.74      4.04      5.95      5.66
    Hi CL mm     1.75     5.12      3.68      5.15      4.83      6.47      6.74
```

Example 43 *Table of growth statistics produced by the TRANSIT module of the GHAFOSIM system* significant correlation with increment and was therefore not used in the model.

TRANSIT Compiles the tree increment data, stored in a compressed binary format, to produce the table of transition frequencies in an internal ASCII format. For each species group, this comprises a model number, followed by two lines giving 5-year transition frequencies for sunlit and shaded trees by size classes. There is then a line of 5-year mortalities, given as percentages.

Example 43 shows the printed output from TRANSIT. The species codes are as shown in Table 3. The species groups are defined within TRANSIT in a series of DATA statements, one for each group, listing the species codes involved. The statistics are compiled for four size classes and two shade classes. For each species group and class, the summary gives the number of sample trees, the percentage annual mortality rate, mean diameter increment in mm/yr, the standard error % of mean increment, and the low and high confidence intervals for increment at P=0.95. The 'Shaded' class includes Dawkins crown exposure classes 1-3, and the 'Sunlit' trees are those with Dawkins scores of 4-5 (see Alder & Synnott, 1992, for exposure class definitions). For the largest trees (≥ 70cm diameter), only a single exposure class is considered.

PREPSIM Compiles inventory data to a stand table, using two steering files to control the process. One defines species group names, model number (corresponding to the transition table), minimum felling diameter, utilization percentage of stems over the minimum diameter, and a list of code numbers for species in the group. The other file defines the run title, working directory, and a list of TSP files to be included. The output table is in an internal ASCII format.

GHAFOSIM Comprises the growth model program itself, about 500 lines of BASIC code.

GHAFOSIM operates on the output files from TRANSIT and PREPSIM to perform projections. The user can specify the felling cycle, the total period of the simulation, the interval in years at which output tables would be printed, and the mode of operation. Two operating modes can be selected:

(1) *Simple stand table projection*. This mode does not simulate logging, but prints a full stand table at each reporting interval showing accumulated growth. Example 44 shows a typical output table in this mode. It is a conventional stand table except that stem numbers are shown for the shaded and sunlit components separately on the lines marked "-" and "+" respectively.

(2) *Simulated logging*. This mode is more complex, and involves splitting the input stand table into a number of copies, one for each felling coupe. Each coupe is then grown independently. The yields produced by the

```
PROJECTED STOCKINGS & VOLUMES AFTER 20 YEARS WITHOUT LOGGING

Species group     Canopy   Trees per square kilometre by cm. diameter classes                 Dmin  N/km2   m3/ha.
                  position 5- 9  10- 29  30- 49  50- 69  70- 89  90-109 110-129 130-149   > 150  cm.  >Dmin   >Dmin
-----------------------------------------------------------------------------------------------------------------------
REDWOODS          -     383     441      64      22
                  +       9      15      94      66      27      10       4       4       1   70     46    5.17
CHLOROPHORA       -      15      21       7       4
                  +               1      13       9       6       3       2       2       2   70     15    2.30
TRIPLOCHITON      -     120     169      62      21
                  +      11      30      84      79      77      60      38      20      15   70    210   29.28
CELTIS            -   1,479   2,523     482      98
                  +       8      17     668     331     116      29       8       2           70    155   13.77
CEIBA             -     167     208      60      12
                  +       9      24      66      37      22      11       7       7      15   70     62   10.82
PIPTADENIASTRUM   -     143     136      33      20
                  +       3       8      47      53      50      28      15       7       6   70    106   13.30
TERMINALIA        -     128     151      40       7
                  +      12      32      46      25      23      22      11       6       4   70     66    9.01
ANTIARIS          -     131     135      22       9
                  +       3       7      38      31      18      14       6       4       5   70     47    6.66
NESOGORDONIA      -     455     338      76      22
                  +       2       8     124      80      29       5                           70     34    2.70
PYCNANTHUS        -     137     130      21       9
                  +               3      36      37      27      16       6       2           70     51    5.41
PTERYGOTA         -     104     106      17       8
                  +       2       5      29      28      21      11       6       3       1   70     42    4.96
GUAREA            -     308     217      32       9
                  +       3       6      39      23       9       3                           70     12    1.01
TURRAEANTHUS      -     110     172      38      16
                  +               1      61      48      21       4                           70     25    2.00
CYCLICODISCUS     -      21      20       5       2
                  +                       7       6       4       5       4       3       6   70     22    4.31
CHRYSOPHYLLUM     -     125     106      17       7
                  +                      28      22      11       6       3                   70     20    2.04
PETERSIANTHUS     -      35      57      16      10
                  +               2      30      32      22      10       5       3       1   70     41    4.73
RICINODENDRON     -     159     133      18       7
                  +       8      19      31      23      11       9       4       3       1   70     28    3.64
OTHER SPECIES     -  40,403  29,544   4,748   1,064
                  +     536     932   5,452   2,813     989     239      64      21      10   70  1,323  119.99
-----------------------------------------------------------------------------------------------------------------------
TOTAL                45,029  35,717  12,651   5,090   1,483     485     183      87      67        2,305  241.09
=======================================================================================================================
```

Example 44 *Output from the GHAFOSIM program running in stand table mode*

forest as a whole are summarised in an output table (see Example 45). The simulated logging removes a percentage of trees above the minimum felling diameter on the appropriate felling coupe for each simulation cycle. The percentage removed (*Util %* in Example 45) can define either the market acceptance of a species, or conservation requirements.

Example 45 *GHAFOSIM output in simulated logging mode*

```
PROJECTED LOG YIELDS PER ANNUM

Felling cycle  25  years :   Total area  11590 km2   :   Annual felling coupe     463.60 km2

Species group   Min.  Util.  <--------------------------- Annual logging extraction per unit area and total --------------------------->
                diam    %          1 -  5 yrs              6 - 10 yrs             11 - 15 yrs             16 - 20 yrs             21 - 25 yrs
                             N/km2 m3/ha '000 m3    N/km2 m3/ha '000 m3    N/km2 m3/ha '000 m3    N/km2 m3/ha '000 m3    N/km2 m3/ha '000 m3
-------------------------------------------------------------------------------------------------------------------------------------------
REDWOODS          70   100        2  0.3     332        2  0.2     290        2  0.2     258        2  0.2     240        2  0.2     233
CHLOROPHORA       70   100        1  0.1     147        1  0.1     124        1  0.1     115        1  0.1     107        1  0.1     101
TRIPLOCHITON      70   100        8  1.0   1,207        8  1.1   1,254        8  1.1   1,304        8  1.2   1,357        9  1.2   1,417
CELTIS            70   100        5  0.4     499        5  0.5     529        6  0.5     578        6  0.6     639        7  0.6     703
CEIBA             70   100        2  0.4     508        2  0.4     492        2  0.4     482        2  0.4     501        3  0.5     526
PIPTADENIASTRUM   70   100        3  0.4     465        4  0.4     519        4  0.5     564        4  0.5     617        5  0.6     676
TERMINALIA        70   100        3  0.3     366        3  0.3     380        3  0.3     394        3  0.4     418        3  0.4     442
ANTIARIS          70   100        2  0.2     258        2  0.2     271        2  0.3     290        2  0.3     309        2  0.3     327
NESOGORDONIA      70   100        1  0.1     102        1  0.1     108        1  0.1     119        1  0.1     125        1  0.1     135
PYCNANTHUS        70   100        2  0.1     171        2  0.2     198        2  0.2     226        2  0.2     251        2  0.2     274
PTERYGOTA         70   100        2  0.2     179        2  0.2     195        2  0.2     214        2  0.2     230        2  0.2     245
GUAREA            70   100        0  0.0      35        0  0.0      40        0  0.0      43        0  0.0      47        1  0.0      50
TURRAEANTHUS      70   100        1  0.1      65        1  0.1      74        1  0.1      86        1  0.1      93        1  0.1     100
CYCLICODISCUS     70   100        1  0.2     175        1  0.2     184        1  0.2     200        1  0.2     200        1  0.2     200
CHRYSOPHYLLUM     70   100        1  0.0      56        1  0.1      69        1  0.1      83        1  0.1      94        1  0.1     109
PETERSIANTHUS     70   100        1  0.1     153        1  0.1     173        2  0.2     193        2  0.2     219        2  0.2     246
RICINODENDRON     70   100        1  0.1     127        1  0.1     139        1  0.1     150        1  0.1     169        1  0.2     191
OTHER SPECIES     70     0        0  0.0       0        0  0.0       0        0  0.0       0        0  0.0       0        0  0.0       0
-------------------------------------------------------------------------------------------------------------------------------------------
ALL SPECIES                      36  4.2   4,844       37  4.3   5,040       39  4.6   5,300       39  4.8   5,614       44  5.2   5,976
===========================================================================================================================================
```

4 COHORT MODELS OF FOREST GROWTH

4.1 Design and structure of a cohort model

4.1.1 *Definition and principles*

The term 'cohort model' is used here to define a deterministic, empirical, management-oriented modelling strategy for mixed tropical forests that has been practically exemplified by the work of Vanclay (1989a) for the North Queensland rainforests. The central features that characterise this type of model are:

- The forest stand is classified into cohorts. A cohort is a group of trees of the same species group, and size class. In a more complex design, the cohorts can also be classified by tree condition, competitive status, and other factors.

- Growth is simulated by updating the parameters defining a cohort over increments of simulated time, which may typically vary from one to five years, depending on model design and perhaps available data. Growth can be considered to involve tree diameter increment, mortality, and recruitment.

- Harvesting is simulated by reducing the numbers in affected cohorts, and perhaps by enhanced mortality proportionate to the harvesting intensity. Where the cohort structure indicates tree vigour or health, harvesting may create a 'damaged' or morbid condition on a proportion of cohorts.

- The spatial heterogeneity that is so characteristic of natural tropical forest is modelled by dividing the whole forest extent into a number of small simulated units, each of limited area. Each simulated forest unit (SFU) is characterised by its own cohort list. For a simple model design, SFUs may correspond to management units such as a compartment or felling coupe. For more precision, the nominal area of the SFUs may be reduced so that each coupe is represented by a number of them.

The main differences between a cohort-oriented model (COM) and a diameter-class projection model (DCPM) are that:

- There is no need to use storage space or processing time on empty classes in a COM. Combinations of species and sizes that are not represented in the data do not create any entry in the cohort database within the model. With a DCPM, on the other hand, empty classes must be represented. This is especially inefficient with tropical mixed forest, where there will be many such combinations of species and diameter class that are void.

- Unless a DCPM is formulated as a transition probability model, assumptions are required about how growth rates correspond to the 'movement' of trees between classes (see page 77). In a COM, the cohort parameter defining size is directly updated by the growth functions, and no such assumptions are needed. This makes the COM more suitable for functional modelling.

Both COMs and DCPMs can characterise diameter growth as a stochastic process, or a probability distribution of growth rates from a single originating class or cohort. In a DCPM, this is performed by considering transition frequencies to other classes. In a COM, the probability distribution of growth rates is modelled by splitting a cohort into faster and slower growing components.

The corollary of a process of splitting cohorts is one of merging them. Two cohorts can become very similar in their defining parameters as a result of differential growth rates. It is then efficient to merge them into a single data structure. This avoids an indefinite proliferation of cohorts as a result of stochastic splitting.

Although COMs may include stochastic elements to represent probability distributions, the outcomes of the model are deterministic in that a simulation with given starting conditions yields the same result for each model run. No Monte Carlo processes, or functions mediated by random number generators, are required. This may be contrasted with models based on representations of single trees, such as gap or tree position models. These intrinsically require a Monte Carlo approach to resolve problems of applying mortality or harvesting functions to individual trees.

On a number of grounds, cohort models may therefore be regarded as the most suitable choice for management-oriented modelling of mixed tropical forest:

- They are efficient in their data structure, and therefore well-suited to the massive and complex data sets that are typical of tropical forests.

- They are inherently suited to a functional approach. It requires less data to characterise a growth function than a transition matrix, and hence this method is more appropriate for modelling tropical forests where many rarer species will have only limited growth data available.

- They are deterministic, which is a desirable, and probably necessary, feature of models that are to be used directly to generate schedules for forest management.

4.1.2 *Programming languages for cohort modelling*

Generally, a cohort model must be written in a programming language capable of being compiled and optimized, as execution speed will be found to be critically important. On machines based on the Intel family of processors (808x6, Pentium), it is desirable for the compiler to be able to generate 32-bit instructions.

During the development stages of a model, ideas can be tested with languages such as FoxPro XBASE, or interpreted Quick BASIC. Some concepts and functions can be tested using speadsheets. However, when a complete model is built using such systems, they will be found to be unacceptably slow.

Suitable programming languages include Microsoft Visual Basic, various versions of C or C+ +, Pascal and FORTRAN. The author prefers C or C+ + as offering maximum possibilities for efficiency and portability between systems. Visual Basic is easier to learn and use, but is relatively slower and less efficient. Some compiled versions of Basic (eg. Microsoft Basic 7) are highly optimized but not convenient to use with the Windows operating system. FORTRAN is familiar to an earlier generation of programmers and is compatible with older software. FORTRAN compilers are available that are highly optimized and efficient. However, FORTRAN is out of step with modern programming paradigms such as the Windows interface or object-oriented programming, and is primitive in many of its language features. Pascal offers most of the advantages of C but without its somewhat terse syntax, and may be considered easier to use.

The CAFOGROM program used as a case study for cohort model design in this chapter is written in Borland Turbo C version 2.0. A demonstration diskette for the program is available from the author. Appendix A gives ordering information. A complete listing of the C code for CAFOGROM is given in Appendix B.

4.1.3 *General outline of a cohort model*

Figure 13 shows the modules or steps that will form part of a typical computer program to implement a cohort model. The references to line numbers in CAFOGROM relate to the listing in Appendix B (page 205 *ff.*)

(i) *Initialization*: The program will include a preamble whose details will be determined by the language used. In C for example, header files for library routines will be declared. Constants, variables, and function prototypes may be defined. Arrays and other data structures used throughout the program will be established. In the CAFOGROM listing lines 1-82 of the program constitute the initialization phase.

(ii) *Data selection and management options*: This part of the program will establish the *user interface*. There are several possible approaches to this. The program may be controlled by numeric and text data entered in *steering files*. This is perhaps the oldest approach. A more modern concept would be to use a combination of menus and screen formats. In a Windows environment, the typical Windows tools of buttons, menus, click boxes, and so would be used. The CAFOGROM example discussed in the case study (page 165) uses a simple screen format. The procedure *RunOpt* at lines 84-169, together with supporting procedures called by it, codes for this user interface.

(iii) *Load external data and coefficients:* This section of the program will read any data it needs from external files. This will always include data describing the forest to be simulated, and may usually include the growth model coefficients fitted during analysis. The forest data may be read in a pre-summarised format, or the program may process directly raw plot

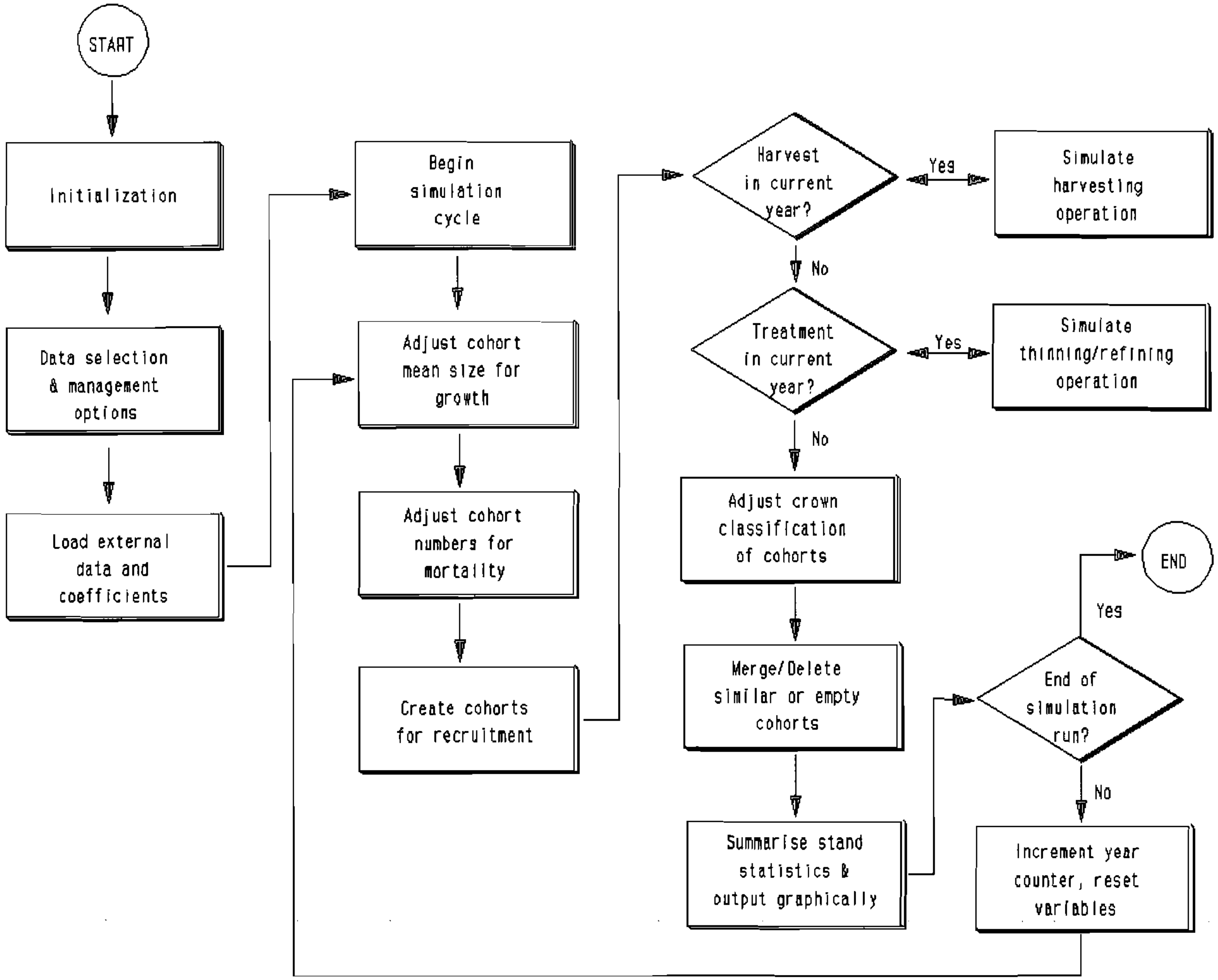

START
Initialization
Data selection & management options
Load external data and coefficients
Begin simulation cycle
Adjust cohort mean size for growth
Adjust cohort numbers for mortality
Create cohorts for recruitment
Harvest in current year?
Yes
Simulate harvesting operation
No
Treatment in current year?
Yes
Simulate thinning/refining operation
No
Adjust crown classification of cohorts
Merge/Delete similar or empty cohorts
Summarise stand statistics & output graphically
End of simulation run?
Yes
END
No
Increment year counter, reset variables

Figure 13 *Flowchart for a typical cohort model*

data down to the required cohort format. In CAFOGROM, procedure *getdata* (lines 252-296) reads forest data, whilst *Setup* (lines 772-852) reads growth function coefficients.

(iv) *Begin simulation cycle:* At this point, the program will have all the information it needs to begin a simulation run. A loop structure will be entered in the program code which will repeat until a pre-set number of time steps have been completed. In CAFOGROM, the simulation cycle is organized as a short procedure called *simulate* (lines 298-334). This calls the more detailed procedures for growth simulation, harvesting and thinning as required.

(v) *Adjust cohort mean size for growth:* At this stage in the program, diameter increment functions are applied to the cohort data structure that represents the forest stand. This is performed in CAFOGROM by the routine *Growth* (lines 411-431).

(vi) *Adjust cohort numbers for mortality:* Each cohort represents a given number of trees. In this part of the program, mortality is simulated by reducing the number of trees in each cohort using functions that may depend on species, stand density, crown position, logging history, or other factors. Mortality is managed in CAFOGROM by a routine of the same name at lines 433-454 in the listing.

(vii) *Create cohorts for recruitment:* The level and species distribution of recruits into the smallest size class is determined from functions that may depend on stand density and past disturbance. New cohorts are created to define the predicted recruitment. In CAFOGROM, the procedure *Recruitment* at lines 456-491 executes these functions.

(viii) *Simulate harvesting*: A test is made to see if conditions are satisfied for harvesting. The test may simply depend on time elapsed (felling cycle), or could embody more complex factors such as stand basal area achieving a required minimum figure. Function *LogCheck* in CAFOGROM (lines 336-354) exemplifies these tests.

If harvesting is required, a procedure will be called that reduces the stocking of cohorts that satisfy the species and size class requirements for trees to be logged. At the same time, damage may be simulated by reductions in numbers of non-harvested trees to reflect tree destruction on skid trails and gaps, or immediate mortality through crown or stem breakage. Longer-term mortality will generally be accounted for in the mortality function by using time elapsed and intensity of previous logging as parameters in mortality functions. Lines 493-563 (procedure *Harvest*) in CAFOGROM implement harvesting simulation.

The harvesting routine will usually summarise statistics on trees removed to make them available at the user interface for display in graphical or tabular form.

(ix) *Simulate thinning or refining operations:* Silvicultural treatments such as crown liberation thinning, understorey cleanings, or refining can be simulated in a way that closely parallels harvesting. Species and size selection criteria may be defined, and cohort numbers reduced for those cohorts which satisfy the required criteria. Generally, damage will be assumed not to occur in such cases, although empirical data may indicate higher mortality for some elements of the residual stand. Thinning treatment is simulated in CAFOGROM by procedure *Thinning* at lines 565-604.

(x) *Adjust crown classification of cohorts*: Not all models will use crown position as a factor, but in those that do (as in the CAFOGROM example), crowns will be reclassified in each simulation cycle, after harvesting and thinning have been completed, to reflect changes that have occurred. In CAFOGROM, this is done by the *CrownAdjust* routine at lines 606-656 of the program.

(xi) *Merge/delete similar or empty cohorts*: Consecutive application of mortality and harvesting functions will lead the stockings of some cohorts to become very small. Generally, some internal factor will be used (such as a stocking of less than 0.01 tree/ha) to delete cohorts whose numbers fall below the predetermined threshold.

In the same way, differential effects of growth and recruitment can lead some cohorts of the same species to become of very similar mean diameter and crown class. In such cases, as described by Vanclay (1989a), cohorts can be merged. The effect of the deletion and merging functions is to keep the number of cohorts required to describe the stand within limits, and avoid a proliferation of data elements during the simulation with little useful information content.

Merging and deletion of cohorts is carried out in CAFOGROM by the routine *MergeCohorts* at lines 687-745.

(xii) *Summarise stand statistics:* At the end of each growth cycle, stand statistics are summarised and may be displayed in a graphical or tabular format to the user. In addition, tabular results will normally be saved in a file for later examination and analysis.

The CAFOGROM demonstration program listed in the Appendix produces only graphical output (several examples are given in Section 5). The procedures from 854-1293 perform these tasks. It will be seen that the output display, although a small part of the program conceptually, embodies a large part of the programming effort. In this case, some 33% of the program code is devoted to output presentation.

(xiii) *Increment year counter, end simulation run:* The cycle of operations from (v) above will be repeated after the elapsed time has been incremented by the time step being used. The time step will usually be in the range from

1 to 10 years, with 5 years being a suitable interval for many situations. When the simulated time reaches some preset limit selected by the user, then the program will halt. In CAFOGROM, this aspect is managed within the *Simulate* procedure (lines 298-334).

These various procedural sections of a program can be subdivided into *logical* and *empirical* components. The logical components are dictated by the details of the program design, and influenced by the programming language used. The empirical aspects consist of the growth, mortality and recruitment functions used. Harvesting, silvicultural treatment and other aspects of stand management are modelled largely on a logical basis, but with some empirical elements to describe damage. The major part of this chapter is concerned with the methods for determining the empirical components of the model.

4.1.4 *Representation of the forest stand as a data structure*

A conventional stand table of species by size classes can be considered as a 2-dimensional array, comprising rows (species) and columns (diameter classes).

A forest stand represented in cohort format should however be seen as a list. Each element of the list consists of a data structure with the following components:

Species (or species group)
Mean size (diameter or basal area)
Crown class
Number of trees per ha.

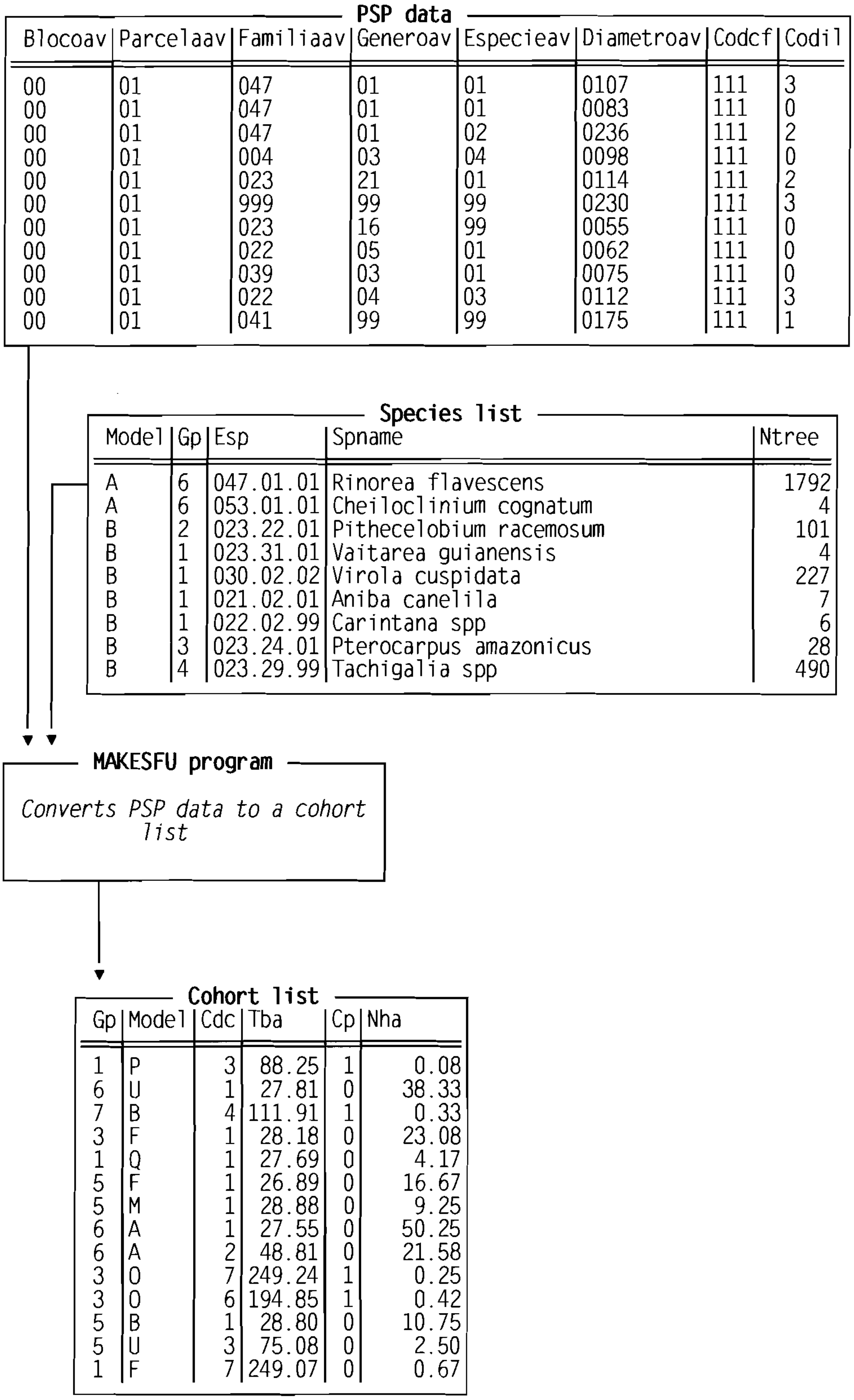
Further components could be added for more complex models, such as damaged versus undamaged trees. However, the reader is cautioned against devising elaborate structures as a basis for simulation. Each variable in the cohort has to be somehow predicted in the model from a growth function: Even species is determined in the recruitment functions. Unless very extensive data is available, adding variables considerably complicates both the analysis and the model.

The list of cohorts representing a single stand may be called a *simulated forest unit*, or SFU. More complex models, which simulate multiple stands, will use arrays of SFUs to represent the forest, with each SFU itself comprising a dynamic list of cohorts. In CAFOGROM, the cohort data structure is defined at lines 53-54 of the program.

Example 46 shows a program called MAKESFU written in Foxpro XBASE that converts PSP data into a cohort list. This program was used on data sets from Tapajos Forest, Brazil (Silva, 1989; Alder, 1994). The data structures on which the program operates are shown in Figure 14. The following is a brief description of how MAKESFU works:

```
* Builds a cohort list in file SFU from PSP data
* Create and index new SFU file
create table sfu (gp c(1), model c(1), cdc n(3), tba n(6,2), cp n(1), nha n(8,2))
select sfu
index on gp+model+str(cdc,3)+str(cp,1) tag cohort
* open species list in file LISPX and PSP data
use lispx order tag esp in 0
use (PSP) in 0 alias PSP
select PSP
np=0
plotid=[xxxx]
* add each live tree to appropriate record of cohort database SFU
scan for CODCF=[111] or CODCF=[112]
  * plot counter update
  if BLOCOAV+PARCELAAV< >plotid
    np=np+1
    plotid=BLOCOAV+PARCELAAV
  endif
  * set group and model numbers
  esp0=FAMILIAAV+[.]+GENEROAV+[.]+ESPECIEAV
  select lispx
  seek esp0
  if found()
    gp0=gp
    model0=model
  else
    gp0=[8]
    model0=[M]
  endif
  select PSP
  * set 2-cm diameter class, TBA and crown class
  cdc0=int((val(DIAMETROAV)-50)/20)+1
  tba0=(val(DIAMETROAV)/10)^2*0.7854
  cp0=iif(CODIL=[0] or CODIL=[3], 0, 1)
  * No/ha conversion factor for 0.25 ha PSPs
  nha0=4
  * find matching cohort
  select sfu
  coid=gp0+model0+str(cdc0,3)+str(cp0,1)
  seek coid
  if not found()
    * add new cohort to file
    append blank
    replace gp with gp0, model with model0, cdc with cdc0, ;
            cp with cp0, tba with tba0, nha with nha0
  else
    * update tba and nha value for existing cohort
    replace tba with (tba*nha+tba0*nha0)/(nha+nha0), nha with nha+nha0
  endif
  select PSP
endscan
select sfu
* adjust stockings for number of plots
replace all nha with nha/np
return

```

Example 46 *FoxPro program to compile a cohort list from PSP data*



PSP data

Blocoav	Parcelaav	Familiaav	Generoav	Especieav	Diametroav	Codcf	Codil
00	01	047	01	01	0107	111	3
00	01	047	01	01	0083	111	0
00	01	047	01	02	0236	111	2
00	01	004	03	04	0098	111	0
00	01	023	21	01	0114	111	2
00	01	999	99	99	0230	111	3
00	01	023	16	99	0055	111	0
00	01	022	05	01	0062	111	0
00	01	039	03	01	0075	111	0
00	01	022	04	03	0112	111	3
00	01	041	99	99	0175	111	1

Species list

Model	Gp	Esp	Spname	Ntree
A	6	047.01.01	Rinorea flavescens	1792
A	6	053.01.01	Cheiloclinium cognatum	4
B	2	023.22.01	Pithecelobium racemosum	101
B	1	023.31.01	Vaitarea guianensis	4
B	1	030.02.02	Virola cuspidata	227
B	1	021.02.01	Aniba canelila	7
B	1	022.02.99	Carintana spp	6
B	3	023.24.01	Pterocarpus amazonicus	28
B	4	023.29.99	Tachigalia spp	490

MAKESFU program

Converts PSP data to a cohort list

Cohort list

Gp	Model	Cdc	Tba	Cp	Nha
1	P	3	88.25	1	0.08
6	U	1	27.81	0	38.33
7	B	4	111.91	1	0.33
3	F	1	28.18	0	23.08
1	Q	1	27.69	0	4.17
5	F	1	26.89	0	16.67
5	M	1	28.88	0	9.25
6	A	1	27.55	0	50.25
6	A	2	48.81	0	21.58
3	O	7	249.24	1	0.25
3	O	6	194.85	1	0.42
5	B	1	28.80	0	10.75
5	U	3	75.08	0	2.50
1	F	7	249.07	0	0.67

Figure 14 ***Data structures used and created by MAKESFU cohort compilation program***

Lines

3-5 Create the output data structure and index it in a way suitable for checking the existence of cohorts. Table SFU is shown as the *cohort list* in Figure 14.

7-9 Opens the existing species list and PSP data files. These are shown as the top two boxes in Figure 14.

10-11 Sets initial conditions and begins a scan through the PSP data file. Only live trees (CODCF of 111 or 112) will be included in the output cohort list. CODCF is a stem code for trees which are broken, leaning, damaged or dead.

15-18 Checks if the current data record is from the same plot as the previous one. BLOCOAV is the block number in which the plot is located, and PARCELAAV is the plot number. If the record is for a new plot, then the plot counter, *np*, is updated.

19-29 These lines convert the species identification format used in the PSP file to the same format as the species list file. This is done by combining the separate codes for family (FAMILIAAV), genus (GENEROAV) and species (ESPECIEAV) into a variable *esp0*. The species list is searched for this (line 22). If a match is found, then the group (*gp*) and model code (*model*) are assigned to the memory variables *gp0, model0*. Otherwise, default values are used (group 8 and model M).

32-36 An entry corresponding to the tree record is created in cohort format. In the PSP file, tree diameter is in mm in character format (DIAMTROAV). This is converted to a 2-cm diameter class index at line 32 (*cdc0*), and to a tree basal area in cm^2 at line 33 (*tba0*). The crown class *cp0* is coded as 0 for understorey trees, or 1 for codominants and emergents at line 34.

Line 36 sets the stocking for the cohort. The plots in this example were of 0.25 ha, meaning that a single tree corresponds to a mean stocking of 4 trees/ha.

37-49 The list of existing cohorts is searched to see if one exists already with the same species group, model, size class and canopy position as that for the incoming tree. If there is already a cohort, then its mean tree basal area and stocking are adjusted for the incoming cohort. Stocking is updated by simple addition. Mean tree basal area is formed as the weighted mean of the current and incoming cohort mean basal areas (line 48).

If no matching cohort exists, a new one is added to the list (lines 43-45).

52-54 After all the data has been scanned, the stocking for each cohort are changed from overall totals to totals per ha by dividing by the plot count.

This program was developed for a specific situation, using the PSP data from

CPATU, Brazil to generate cohort files for use by the CAFOGROM model. However, by amending the field names used and the details of the species coding and look-up method, it can be readily adapted to compile cohort lists from permanent or temporary plot data in any type of DBF-compatible file format.

4.1.5 *Steps in developing a cohort model*

A complete cohort model will be developed through a number of stages. This will usually commence with the analysis of diameter increment and mortality functions. During this process, it will be necessary to devise some scheme of species grouping that reduces the large number of species found in tropical mixed forest to a limited number of sets. Vanclay (1994:127) discusses some possible strategies for this.

Recruitment functions will be required, and can be expected to present a more difficult problem than increment and mortality. Recruitment will depend very much on levels of past disturbance.

Once the increment, mortality and recruitment functions have been provisionally determined, a complete model can be tested, which should show some of the properties discussed in the section on model validation. During this testing stage, it is likely that weaknesses will be seen that will require the original functions to be revised on some degree.

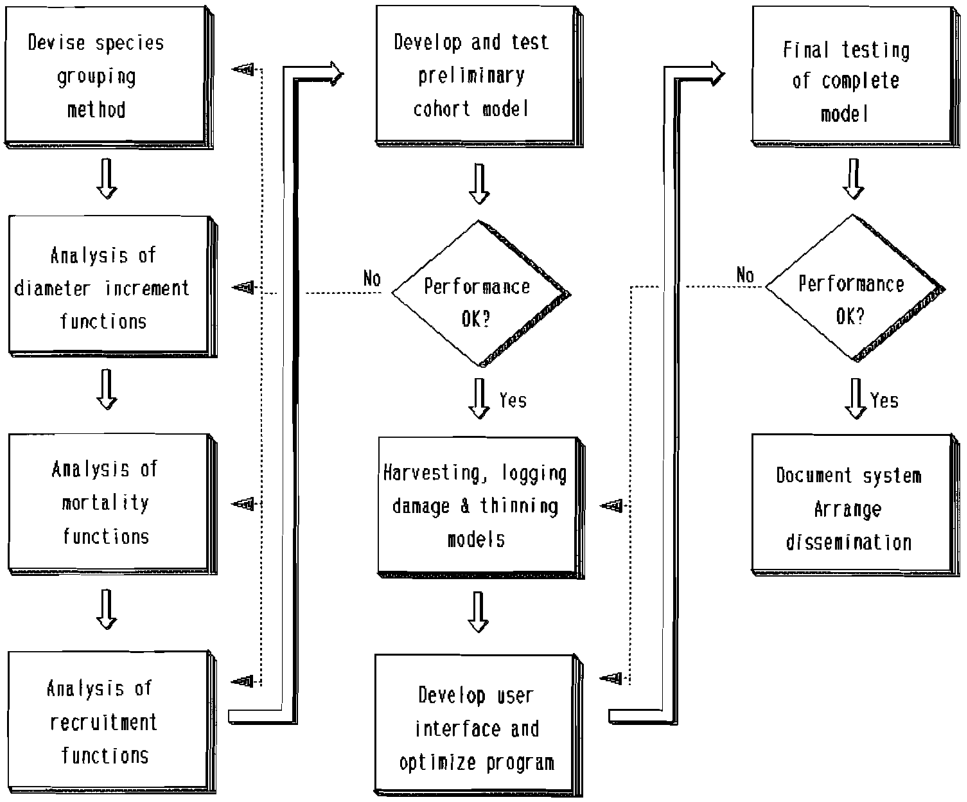

Devise species grouping method
Analysis of diameter increment functions
Analysis of mortality functions
Analysis of recruitment functions
Develop and test preliminary cohort model
No
Performance OK?
Yes
Harvesting, logging damage & thinning models
Develop user interface and optimize program
Final testing of complete model
No
Performance OK?
Yes
Document system Arrange dissemination

Figure 15 ***Steps in the development of a cohort model***

The model can then be elaborated to include harvesting and thinning techniques, with further testing of these components. The user interface can be improved at this stage. The remainder of section 4 details some of the possible approaches to developing functions for increment, mortality, recruitment and harvesting.

4.2 **Tree increment functions**

4.2.1 *Basic considerations*

A predictive function for tree diameter increment is fundamental to a cohort growth model, as it is to other functional models based on individual trees or size classes. Vanclay (1994, Chapter 8) reviews the many approaches that have been developed for diameter increment equations.

In mixed tropical forest, diameter increment may be predicted empirically from tree diameter or basal area, competitive status of the tree or stand, and continuous or categoric site variables. Other variables such as tree age and height are not easily measurable in mixed tropical forest, and are therefore not usable in models for management purposes, although they may be of interest for research studies.

It is often more satisfactory to develop predictive models for tree basal area increment than for diameter increment. It is easier to devise linear forms, higher correlations are found (eg. West, 1980; Foli, 1993), and tree basal area provides for more economical algorithms within the cohort model. However, this is in the end largely a matter of personal preference, and examples of both approaches are given in this section.

The symbolic names for diameter and basal area increment, and the common regression variables used to predict them, are shown in Table 4. The principle dependent variables are DI (diameter increment) or BAI (basal area increment). The predictor variables are tree basal area (TBA) or diameter (DBH), stand density usually measured as stand basal area (SBA), overtopping basal area (OBA) as an index of competition, the crown position, frequently scored using the Dawkins crown classification (see Alder & Synnott, 1992) and tree maximum diameter.

Other predictor variables for regressions can be envisaged, but for a model that is to be applied to forest management they should satisfy an essential criterion: Can they be routinely determined as part of forest inventory operations ? For example, crown projection area might be found to be a good predictor of increment. However, it cannot be directly measured on any substantial sample of the forest, and is therefore of limited value as a predictor in practical applications.

The relation between diameter increment (DI) and tree basal area increment (BAI) can be derived geometrically from:

DBH	Tree diameter at breast height or other reference height, in cm.
TBA	Tree basal area at the reference height, in cm^2.
DI	Tree diameter increment, or annual change in DBH, in cm/yr.
BAI	Tree basal area increment, or annual change in TBA, in cm^2/yr.
SBA	Stand basal area, in m^2/ha.
OBA	Overtopping basal area, in m^2/ha. The basal area of all trees in the stand larger than a specified DBH.
DCC	Dawkins Crown Competition index. A code from 1-5 for crowns ranging from completely shaded in the understorey to those emergent or growing freely in large gaps. See Alder & Synnott (1992) for more details.
SQn	Site quality categoric codes, where n may be 1-9. These variables take the value 1 for sample plots located on the specified site type, and 0 for plots located elsewhere. SQ1 does not necessarily imply faster growth than SQ2, etc. for any given species.
DMAX	The largest observed diameter tree for a given species, cm. This may be defined in absolute terms, without reference to sample area, but is more meaningful as the largest observed per km^2 or other spatial extent, or a percentile (eg. 99%) of the overall diameter distribution of the species.

Table 4 *Symbolic names of variables used in diameter and basal area increment regressions*

$$\begin{aligned} \text{BAI} &= \tfrac{1}{4}\pi[(\text{DBH}+\text{DI})^2-\text{DBH}^2] \\ &= \tfrac{1}{4}\pi[2.\text{DBH}.\text{DI} + \text{DI}^2] \\ &\approx \tfrac{1}{2}\pi\text{DBH}.\text{DI} \end{aligned} \tag{37}$$

The DI^2 term is generally negligible and can be disregarded for an approximation (it will tend to zero as the measurement interval becomes shorter). For example, a 30cm tree with 0.5 cm increment will have 2.DBH.DI of 30 cm^2 and DI^2 of 0.25 cm^2. However, for periodic measurements (eg. 5-year intervals), DI can become a substantial fraction of DBH for smaller trees and the DI^2 term should be calculated.

As a rule of thumb to compare diameter and basal area increments, basal area increment in cm^2/yr will be about 1½ the product of diameter and diameter increment.

When increment data from mixed tropical forest are examined, they will be found to contain a high proportion of very small increments and a strongly skewed distribution. As discussed in section 4.2.7 in more detail, this distribution closely approximates the Weibull distribution (Bailey & Dell, 1973). For regression analysis, homoscedacity (uniform variance) about the fitted line is assumed in the mathematical model. It can be approximately achieved with increment data by applying a log transformation.

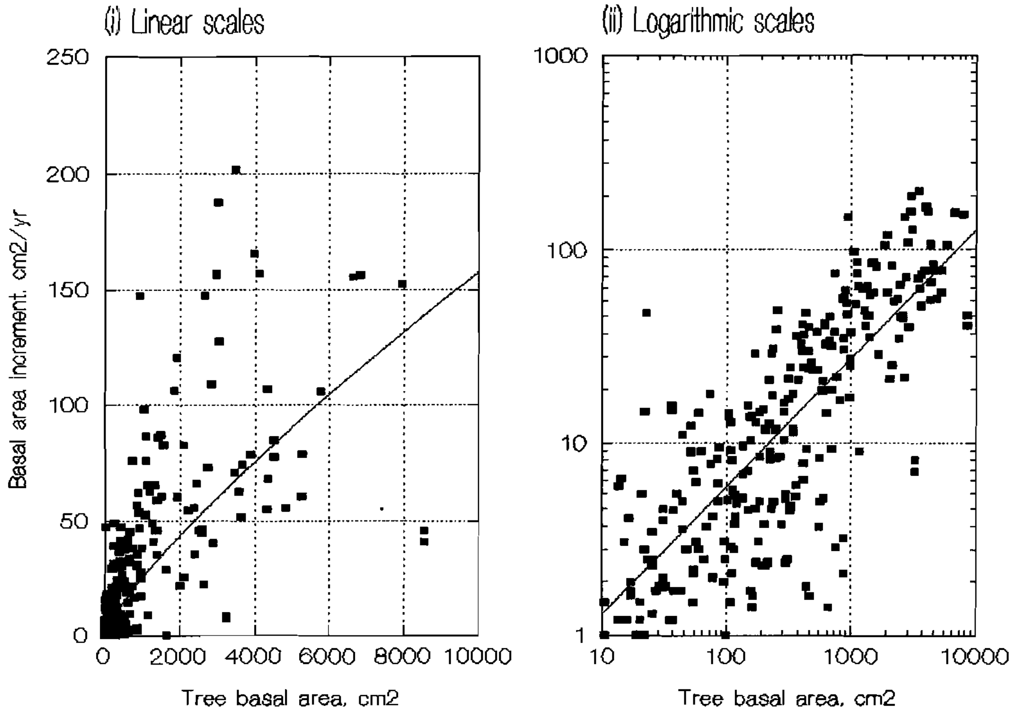


Figure 16 *Effect of log transform on typical increment data (Khaya ivorensis, Ghana)*

This point is illustrated in Figure 16, which shows a typical set of basal area increment data, for *Khaya ivorensis* from permanent sample plots in Ghana (Alder, 1990). On the left, in Figure 16(i), the data are shown plotted directly, BAI on TBA. It will be seen that there are many observations on small trees having low increment, and rather few on larger trees. It is also apparent how the variability of the increment increases with tree size. This may be partly a function of the increasing propensity for measurement errors as the trees get larger.

In Figure 16 (ii), logarithmic axes are used. On the X axis this tends to extend the scale on which the small trees are plotted, and compress the scale for the larger, giving a more even distribution of the sample. On the Y scale, the proportional increase in errors becomes a uniform function, as is required for the correct application of regression theory.

It is recommended that for these reasons a logarithmic transformation of increment data is used, and the dependent variable fitted is either log(DI) or log(BAI). In some cases, the achievement of a linear form may require a more complex transformation of the dependent variable. The direct fitting of DI or BAI will rarely give a good result to ungrouped data, although it can be used, as will be discussed, for means of data points grouped into classes.

The independent or predictor variables also need to be transformed, again principally by taking logarithms. This does not effect the error structure of the fitted regression, but does give suitable linear shapes.

The same effect on the weighting and shape transformation of the regression is achieved irrespective of the base used. This will conventionally be either e (natural logarithms) or 10. Mathematically, and with most computer packages, $\log_e(x)$ is an easier quantity to work with than $\log_{10}(x)$ and it is assumed here that unless otherwise qualified, all logs are to base e. On the other hand, for graphs base 10 or even base 2 logs will give clearer axes with integral tick values. This will depend to some extent on the mechanism involved in the particular graphics package used. The equivalences shown in Table 5 are useful for reference and for conversions.

$\log_{10}(x)$	=	$\log_{10}(e).\log_e(x)$
$\log_{10}(e)$	=	0.43429...
e	=	2.71828...
a	=	$\log_e(x)$
x	=	$\exp(a)$
$\log(1)$	=	0
$\log(0)$	=	$-\infty$

Table 5 *Useful equivalences for log transforms*

4.2.2 *Organizing and transforming data for analysis*

In section 2, methods are described for producing files of diameter increment data from separate data files for each plot assessment, or from files arranged in tabular format, with diameter measurements for each tree arranged in sequence on the same record.

The example program XOBA (page 53) shows how BAI, TBA, SBA and OBA can be calculated from data in columnar format.

Program XTAB (page 44) shows the derivation of grouped mean diameter and mean increment data in output files suitable for analysis.

The files for regression analysis should be constructed from the primary PSP database files using simple programs such as XOBA or XTAB. The output files should contain:

- Key identification variables for the plot, tree and measurement year. These will probably not be used in the regression, but are needed to cross-check outlying points or other features of interest. For classified data, these variables do not apply.

- The species, coded as a categorical (text) variables. If numeric values are used, then they should be stored in a character-type field in the XBASE file. It may be preferable however to use a mnemonic code, as this assists in labelling the output from the statistical analyses.

- One or more variables coding the site factors. These will be categorical (text) variables for vegetation type or forest association, soil group or sub-types, and perhaps codes for local assessments of forest condition. In some cases there may be a known ordinal scheme of site classification, in which case numeric values can be used instead of or as well as categoric variables.

- The key regression variables. Depending on whether diameter increment or basal area increment are to be fitted, or indeed both are to be tested, this may include DI, DBH, BAI, TBA. It will also include the stand density variables SBA and OBA. If a tree position-related index of competition has been calculated, it may be included as CIO, etc (see section 2.6.2(iii)). If the data has been classified, these variables should be class means, and not class mid-point values, as they can differ considerably, and the mid-point value is indeterminate for the largest class.

- If classified data are used, additional variables should give the number of trees in each class, and the variance, standard deviation or standard error of the mean increment observation. These allow weighted regressions to be fitted, using the inverse of the class variance, square root of tree numbers or some similar weighting scheme.

The file should be physically sorted, not simply indexed, on species. It is suggested that a temporary version of the file is created from the summary program (such as XOBA), which is then indexed using species+plot+tree+measurement year and copied to a second permanent file. Copying an indexed file in XBASE is equivalent to sorting it with SORT but is generally faster. The output file will have a physical record sequence corresponding to the logic sequence established by the index.

The data to be processed may be either raw increment measurements, one record per tree, as in the example output file from XOBA (page 53), or classified data in which each record represents the mean increment for a size and competitive status class (as in the XTAB output).

Classified data is more compact but requires some prior knowledge of the most appropriate variables, and also is complex to apply when there is a 3 or 4-way classification, as may happen when there are more than one kind of site factor. Classification is most appropriate when only a simple diameter increment/diameter function is required, as in the XTAB example.

The transformed variables required to create particular linear forms of equation will depend on the equations involved. For the various examples discussed in the following sections, the main transformed variables and their symbolic names are listed in Table 6. The definition and units of the original variables are shown in Table 4.

These transformations include:

- Log transforms of diameter, diameter increment, basal area and basal area increment.
- For some types of functions to be fitted (eg. equations (40) and (43) discussed later) the variables (DMAX-DBH), log(DMAX-DBH), and log(DI/(DMAX-DBH)) will need to be constructed.
- Various competition indices derived from stand basal area (SBA) and overtopping basal area (OBA) will be required. That illustrated in Table 6 is called TCI (Tree Competition Index), derived as (1-OBA/SBA).

Symbol	Definition
Dependent variables:	
LDI	log(DI)
LBAI	log(BAI)
LDIDXD	log(DI/DXD)
Predictor variables:	
LDBH	log(DBH)
LTBA	log(TBA)
DXD	(DMAX-DBH)
LDXD	log(DIDX)
TCI	(1-OBA/SBA)
LTCI	log(TCI)

Table 6 *Symbolic names of transformed variables in increment regressions*

The best approach to generating these variables will depend on the statistical package used. In the case of SYSTAT and similar integrated statistical systems, it is more convenient to construct the transformed variables within the statistical files. For other cases, it may be simpler to do so within the database package used to store the PSP data.

Example 47 shows the commands to create the transformations described in SYSTAT. The user enters the DATA module and either types the commands listed, or runs them in a block as a command file. The / SINGLE qualifier on the SAVE command creates as single precision data file, which is adequate.

The IMPORT command imports the XBASE file XOBA.DBF. This file is decribed in section 2.6.1 and contains the variables GENSP (species mnemonic), BAI, TBA, SBA and OBA.

```
save temp / single
import spoba / type=dbase
use temp
save spoba / single
if bai>500 then let bai=.
let lbai=log(bai)
let lsba=log(sba)
let loba=log(oba)
let ltba=log(tba)
let tci=1-oba/sba
let ltci=log(sci)
run
```

Example 47 *Command file for Systat DATA module to create transformed variables for increment regressions*

The IF statement screens out increments larger than 500 cm^2/yr by setting them as missing values. The LET statements construct the various transformations, which are executed in a batch using the RUN command.

In SYSTAT the EDIT module can also be used to create transformed variables, using very similar commands. However the transformations are then executed immediately with each LET statement, rather than being batched together and performed as a group. This is very much slower with large files.

With limited data sets, the transformations, graphical analysis and regressions can also be conveniently done within a spreadsheet program such as Lotus 1-2-3. For such applications, it is recommended that separate files are created for each species, in order to:

- reduce the file size to proportions manageable within a spreadsheet; and
- avoid the need to make selections based on species within the spreadsheet. This latter operation is very awkward to perform in such packages.

It is a simple matter to use a COPY command in XBASE to selectively copy data from a master file for only a single species. A set of commands such as:

```
USE INCRDAT ORDER TAG GENSP
FIND KHAIV
COPY TO KIDAT WHILE GENSP=[KHAIV]
```

would copy records from a master file INCRDAT, indexed on the field GENSP for all records where GENSP is "KHAIV". The FOR clause can be used on the COPY command instead of while but is significantly slower when working with large data sets as it does not use the index to access required records.

4.2.3 *The shape of the diameter increment function*

Figure 17 shows the typical shapes encountered for the diameter increment and basal area increment function. The data points are mean increments for 10cm diameter classes, produced using the XTAB program described in section 2.5.7. Classified data of this kind shows the shape of the curves more clearly, as will be apparent from a comparison of Figure 17 with Figure 16.

The data is from PSPs in Ghana (Alder, 1990). Three species commonly occurring in West African mixed tropical forest are shown: *Triplochiton scleroxylon* (Obeche/Wawa), a common and ubiquitous fast growing species of the natural forest; *Khaya ivorensis* (Afican Mahogany), a high-valued species of moderate growth rate, and *Guarea cedrata* (Guarea), a slower-growing, smaller tree that is nonetheless common and valuable for timber.

For each species diameter increment means are plotted on the left, and basal area increment on the right. The lines shown are drawn using a spline function* and do not represent any specific growth function; they are intended to give an impression of the shape of the mean line.

The general shapes indicated by these curves rise from close to zero for the smallest trees to a maximum in the mid-size range, and then decline again towards a theoretical zero increment for the largest possible tree. The size of large trees in forest inventory data sets can yield estimates of this intercept point (DMAX), which is a species-dependent parameter. Many growth models of mixed species stands have used DMAX to define an upper limit to tree growth (eg. Botkin et al.,

* This is achieved using the SMOOTH=DWLS option on the SYGRAPH plot command in SYSTAT.

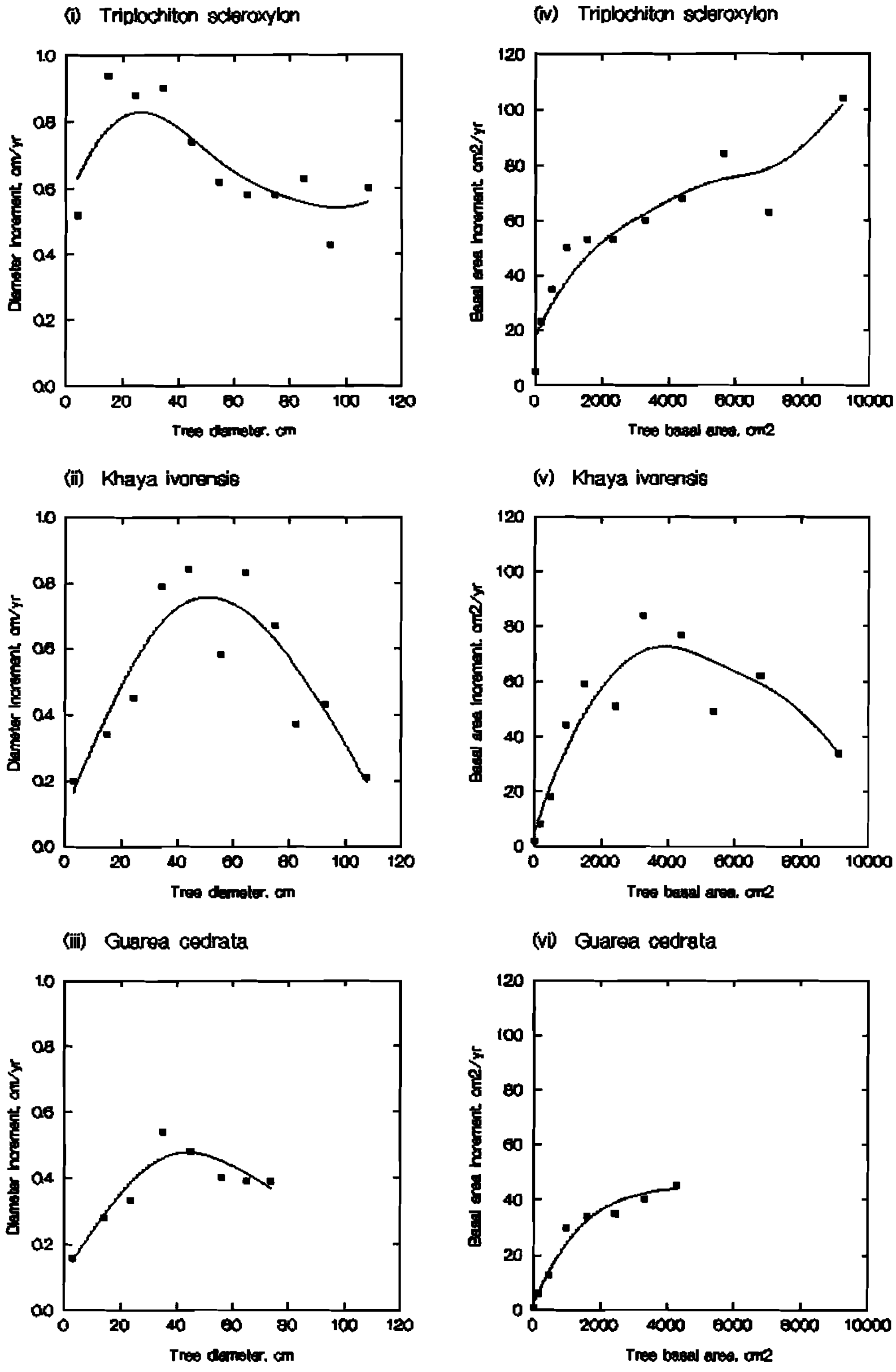


Figure 17 *Diameter and basal area increment functions, illustrated for three common West African species*

1972; Ek & Monserud, 1974; Vanclay, 1989a).

The basal area curve is similar to that for diameter but is skewed to give a sharper rise on the left-hand portion of the curve, and slower decline on the right hand.

4.2.4 *Robust linear models for diameter and basal area increment*

Vanclay (1994) reviews many of the empirical equations that have been used to describe these type of shapes in tree growth studies. The general forms that have been most widely used are the following:

(i) Quadratic

$$DI = a + b.DBH + c.DBH^2 \qquad \{38\}$$

Examples: Alemdag(1978); Rai (1979); West (1980)

(ii) Power-Exponential

$$BAI = a.TBA^b.\exp(c.TBA) \qquad \{39\}$$

Examples: Zeide(1990); Wykoff (1990)

(iii) Modified Beta

$$DI = a.(DMAX - DBH)^b.DBH^c \qquad \{40\}$$

Examples: Vanclay (1989a)

The symbols DI, BAI, DBH, TBA, DMAX are as previously defined. Coefficients a,b,c are estimated by regression. Equation {39} has been generally applied to basal area increment, but can also be used for diameter, although its shape is less suitable for this.

The quadratic function {38} can be fitted directly by linear regression. It fits most sets of tree increment data reasonably well, but is a dangerous function to use because of its inherent flexibility (Figure 18). Only the form with a positive b-coefficient and negative c are suitable, with an a-value that is positive and close to zero. Sets of data from tropical forest PSPs are characterised by many species, most of which are represented by only a few data points. Under such circumstances, the 'best fit' curve will frequently be one characterised by a c coefficient greater than 1, leading to an increment function that ascends without limit, and a model that can generate trees of potentially infinite size.

Equation {39}, termed here the Power-Exponential function, is most usually used to fit basal area increment to basal area. Its shape is probably more appropriate for this purpose than for fitting diameter increment. The linear form is obtained

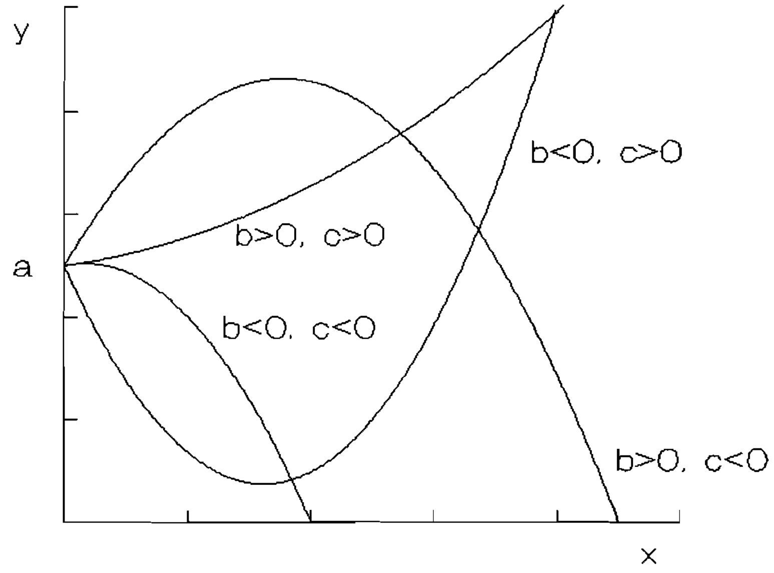


Figure 18 *Shape of the quadratic function with different coefficient values*

by taking logarithms of both sides, giving:

$$\log_e(BAI) = \alpha + b.\log_e(TBA) + c.TBA \qquad \{41\}$$

The coefficient a in {39} is obtained as exp(α). Figure 19 shows the function fitted to data for *Triplochiton* and *Khaya*; the graph shows the same data as Figure 17, but includes error bars for each point. These are the standard errors of the class means as calculated by XTAB (see page 44). Provided the c coefficient is negative, the predicted increment will tend to zero as TBA becomes larger. The b-coefficient should be positive, giving zero increment for trees of zero basal area; with a negative b, increment will be infinite for a tree of zero size.

The modified Beta function is similar in form to the Beta function (Zohrer, 1970; Loetsch et al., 1973:48)*:

$$y = a.(v\text{-}x)^b(x\text{-}u)^c \qquad \{42\}$$

The function is defined from a minimum of x=u to a maximum of x=v. The modified function proposed by Vanclay (1989a) uses a minimum u=0 and a maximum v=DMAX. The power term b to (DMAX-DBH) is assumed to be unity. This gives a robust function which can be fitted to even quite small data sets to give an increment function of sensible shape. The point is illustrated in Figure 20 with the *Guarea* data. Vanclay's model (equation {40} with b=1) and the more general

* The symbols a,b, α, ß used by Zohrer and other authors for the Beta function have been changed here for consistency with other equations in the text.

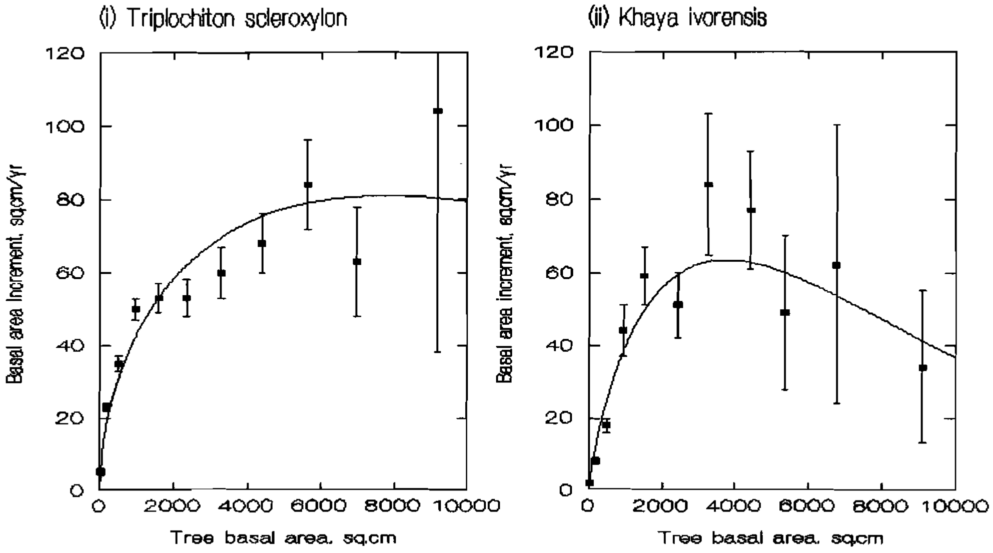


Figure 19 *Power-exponential function fitted to basal area increment data*

form with b estimated from data have been fitted, using the linearized forms {43} and {44}. Both equations fit the data reasonably well, but the less constrained form tends to infinity at DMAX, rather than zero, due to a fitted negative b-coefficient.

Vanclay's function can be linearized by dividing both sides by (DMAX-DBH) and taking logarithms, giving a two-parameter regression equation:

$$\log(DI/(DMAX-DBH)) = \alpha + c.\log(DBH) \qquad \{43\}$$

The a-coefficient in equation {40} is exp(α). The more general form of equation {40} can be converted to linear form simply by taking logarithms of both sides:

$$\log(DI) = \alpha + b.\log(DMAX-DBH) + c.\log(DBH) \qquad \{44\}$$

It will be noted that the dependent variable in equations {43} and {44} differ and they cannot be compared directly for goodness-of-fit using the coefficient of determination, R^2. The transformation tends to weight the observations in different ways. With the pure logarithmic transform, as in equation {44}, lower weight is given to larger diameter trees as compared with the smaller ones. Equation {43}, on the other hand, will give a high weight to values close to DMAX. This is the principle reason why, in Figure 20, Vanclay's equation gives a more reasonable curve.

Some authors have used nonlinear functions derived especially from the Bertalanffy equation (Bertalanffy, 1957):

$$DI = aDBH^b - DBH^c$$

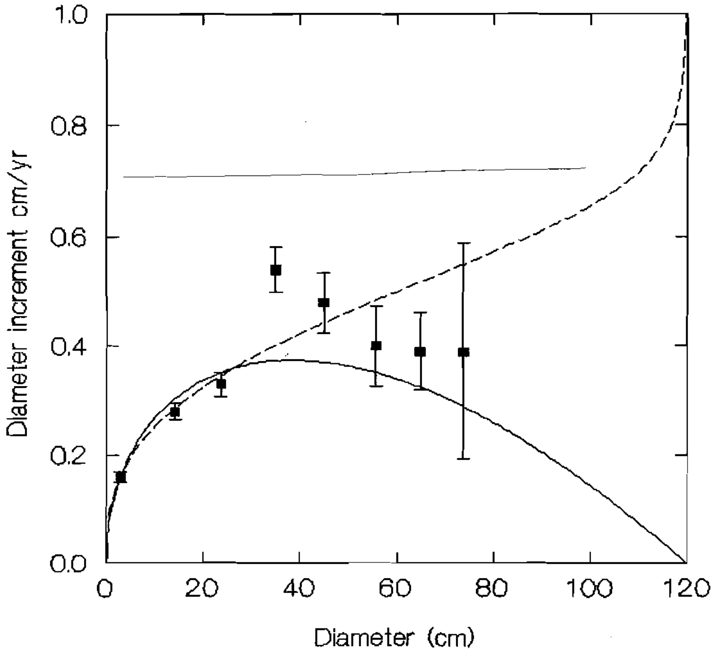


Figure 20 *Vanclay's (———) and the modified Beta function (- - -) fitted to* *Guarea cedrata* *data*

Various derived forms of this and other nonlinear functions are discussed in Vanclay (1994). However, there are practical difficulties in applying such models to tropical forest increment data, and they are better avoided. Some of the possible pathologies that may arise with quite simple linear models have been seen. With nonlinear models these problems are compounded by considerable difficulties in fitting equations when the data is sparse or distributed in a manner that tends to contradict the basic shape of the function.

4.2.5 *Analysis of increment data with competition effects*

In constructing a growth model, the ideal is perhaps a set of diameter increment equations that are so defined that growth on a tree will be zero at levels of stand density corresponding to maximum basal area. In other words, the model is self-limiting in its growth.

In practice this situation does not precisely arise in a mixed tropical forest, and it may not be easy to define such an equation. Diameter increment (growth) and mortality are both active processes, and even at high basal areas, trees are both dying, creating gaps, and growing in the space provided. Emergent trees, free of competition, never appear to cease from adding increment; they simply reach a point where they are blown down as they are weakened by internal decay. The equilibrium of the stand is a dynamic and fluctuating one, composed of a balance between tree increment, mortality, and ingrowth.

A further problem that will be encountered when fitting increment models including competition indices is that there tends to be a strong correlation between competitive status and tree diameter. Consequently, once an equation relating increment to tree diameter has been fitted, such as those discussed in the

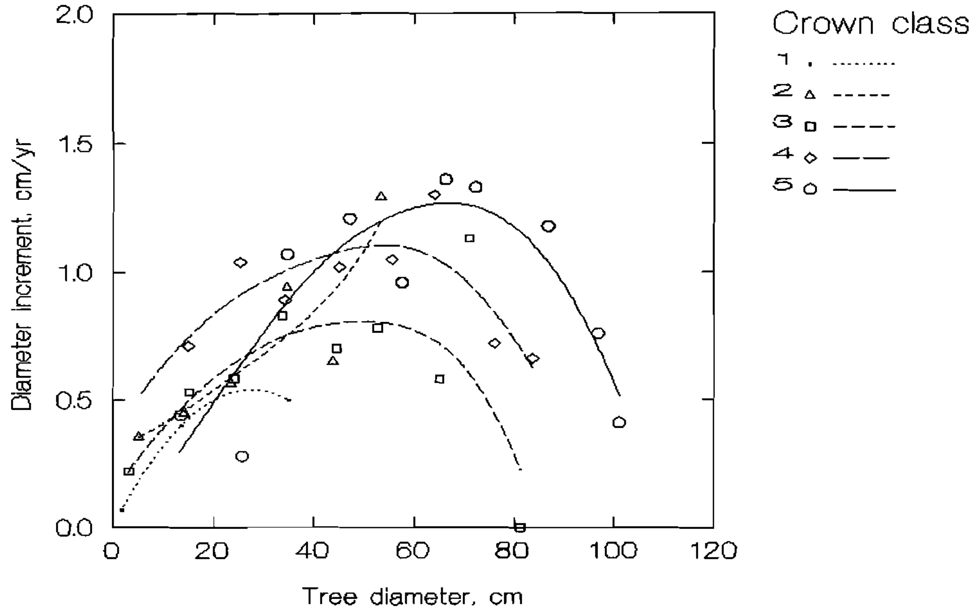


Figure 21 *Diameter increment of* ***Piptadeniastrum africanum*** *(Dahoma) grouped by diameter and crown classes*

preceding section, only a little further variation may be explained by the competition index.

Figure 21 shows a typical situation. Tree increments for *Piptadeniastrum africanum* from PSPs in Ghana have been grouped by 10 cm diameter classes and Dawkin's crown codes in a 2-way classification, using a program similar to XTAB (see page 44). The different classes have each been fitted with a smoothed curve, using the SYSTAT SMOOTH=DWLS function, to illustrate the general trend. The appearance is of a function similar to Vanclay's model, equation {40} with b=1 (Vanclay, 1989a). There appears to be a set of nested curves, with the highest crown class (least competition) showing the greatest increments.

This data was fitted to a single regression function using the following procedures:

- A suitable value for DMAX was determined. For this example a somewhat arbitrary figure of 120 cm was used, but in a real case, a large set of inventory data should be scanned to determine a true maximum for the species*.

- The derived variable log(DI/(DMAX-DBH)) was constructed. In SYSTAT the DATA or EDIT modules will perform this function, using commands such as:

```
LET DMAX=120
LET LDIDXD=LOG(DI/(DMAX-DBH))
```

* JK Vanclay (personal communication) notes that a better procedure might be to allow DMAX to be a function of crown position, as for example, DMAX=45+15*CPOS.

- The derived variable log(DBH) was constructed as a variable LDBH.

- Equation {43}, which is the linearized form of Vanclay's model, was fitted initially to determine how good a fit would be obtained without considering competition effects. The result is shown in Figure 22 (i) as a straight line through the data points. This equation was fitted in the SYSTAT MGLH module using the commands:

  ```
  MODEL LDIDXD=CONSTANT+LDBH
  ESTIMATE
  ```

  In other words, a simple linear regression in the transformed variables was fitted. The fitted function was:

  $$\text{LDIDXD} = -7.6542 + 0.8635\ \text{LDBH} \qquad ; R^2=0.865$$

- Examining this line with respect to the data, it was considered that this most efficient result would be obtained with sets of parallel curves whose intercept depended on crown status, CPOS. CPOS was therefore introduced into the regression in the form:

  ```
  MODEL LDIDXD = CONSTANT+CPOS+LDBH
  ```

  The fitted regression model was:

  $$\text{LDIDXD} = -7.743 + 0.0898\ \text{CPOS} + 0.8011\ \text{LDBH} \ ; R^2=0.898$$

- The fitted model was then transformed back into its nonlinear form in order to examine its shape graphically. The nonlinear form is:

  $$\text{DI} = \exp(-7.743+0.0898\ \text{CPOS}).(\text{DMAX-DBH}).\text{DBH}^{0.8011}$$

  This form has been plotted in Figure 22 (ii) for values of CPOS from 1 to 5.

A further step, which is not illustrated here but which should be undertaken, is to examine the residuals from the back-transformed function against DBH or CPOS for bias or lack of fit. This need not be done for every species, but is desirable at an early stage in function development and testing for those species with large numbers of observations.

The function fitted in this case was an extension of Vanclay's model to allow for crown class:

$$\text{DI} = \text{a}.\exp(\text{b}.\text{CPOS}).(\text{DMAX-DBH}).\text{DBH}^{\text{c}} \qquad \{46\}$$

This in its linear form gives sets of parallel lines whose intercepts depend on the crown position. Similar functions can be used with other competition indices, such as stand basal area, overtopping basal area, or derived indices.

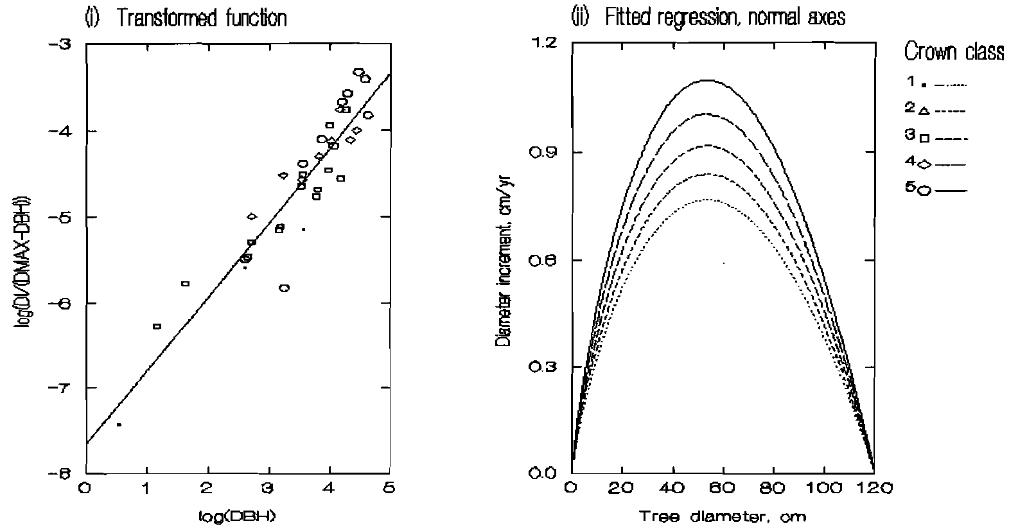


Figure 22 *Linearized form and fitted function for increment model of Piptadeniastrum africanum*

When data is used from PSPs which represent only a narrow spectrum of stand densities, then derivation of functions which include a competition index that behaves in a logical manner may be difficult. This is illustrated for the case of two PSPs at Kade, Ghana managed by the University of Ghana, Legon. The background to the plots is presented by Swaine et al. (1987), and their population dynamics considered in Swaine (1992). The plots are both of 1 ha, and are unlogged. For measurement purposes they are divided into 8 subplots, whose basal areas over time are shown in Figure 23. With the exception of one subplot influenced by a large tree fall, all have high and stable basal areas, with an overall mean of 30 m²/ha.

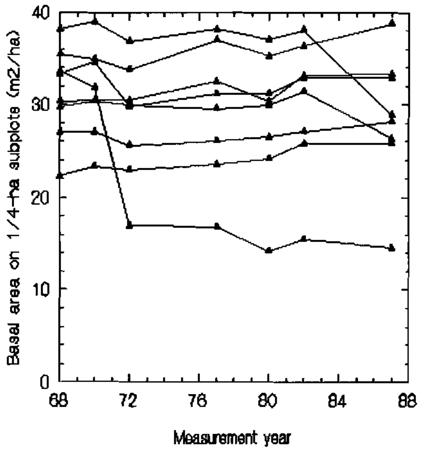


Figure 23 *Basal area on Kade plots, 1968-87*

A function relating basal area increment (BAI) to tree basal area (TBA) and a competition index derived from overtopping basal area (OBA) was fitted for several species. OBA was defined as the sum of the basal areas of all trees larger than a given subject tree on a subplot. The competition index, TCI, was defined as:

$$TCI = 1 - OBA/SBA \qquad \{47\}$$

This has the property that for the smallest tree on the subplot, it will be close to zero, whilst for the largest it will be 1. OBA does not include the subject tree's own basal area, and hence OBA must always be less than SBA, and TCI greater than zero. This is relevant when making log transforms, as log(0) cannot be calculated.

It was desired to fit to the Kade data a model similar to equation {39}, but incorporating the effects of competition as indicated by the TCI index. Data for *Piptadeniastrum africanum*, selected from among the many species on the plot, was used as an indicative test set. The best predictive equation was chosen from among a number of possible variants using stepwise regression. It was:

$$\text{Log(BAI)} = -9.976 - 0.540\log(\text{TCI}) + 1.958\log(\text{TBA}) - 0.000399\text{TBA} \qquad \{48\}$$

The R^2 of this function, fitted to 82 observations, was 71%. It will be noticed that the coefficent of the log(TCI) term is negative, implying that as the competition index increases (ie. the tree becomes more dominant in the stand), the growth rate will be reduced. This is contrary to common sense and experimental evidence, and is not an acceptable result for a growth model. It arises because the observational data does not span a wide range of stand densities, and there is a close correlation between the competition index and tree size. Large trees have higher growth rates (because they tend to be dominant), and also tend to occur on plots of high basal area (because they contribute a large component to the basal area). The result is that with a narrow range of stand densities represented, contradictory effects may well arise when trying to determine the effects of competition on growth.

As a matter of pragmatic necessity it may nonetheless be required to produce a reasonable growth model from data that is not optimal. This can be approached by selecting a function which is not necessarily the best fit, but behaves in a logical manner. For example the simple regression:

$$\text{LBAI} = 0.900 + 4.894\ \text{TCI} \qquad \{49\}$$

gave an R^2 of 56%. This shows the required robust property of increasing growth rate with decreasing competition, and would allow a model based on it to show increased increments following a reduction in stand density. A further study of variants lead to an equation:

$$\text{LBAI} = 6.204\ \text{TCI} - 2.510\ \text{TCI}^2 + 0.162\log(\text{TBA}) \qquad \{50\}$$

This equation is shown plotted on the data in Figure 24. Although there is some lack of fit, the function tends to slightly conservative estimates of increment, and appears stable in its features.

4.2.6 *Site effects in increment functions*

The question of how to incorporate site variations into mixed tropical forest growth models has no simple answer. Vanclay (1994, chapter 7) reviews some of aspects of the problem in detail. In this section, the various practical approaches are considered briefly.

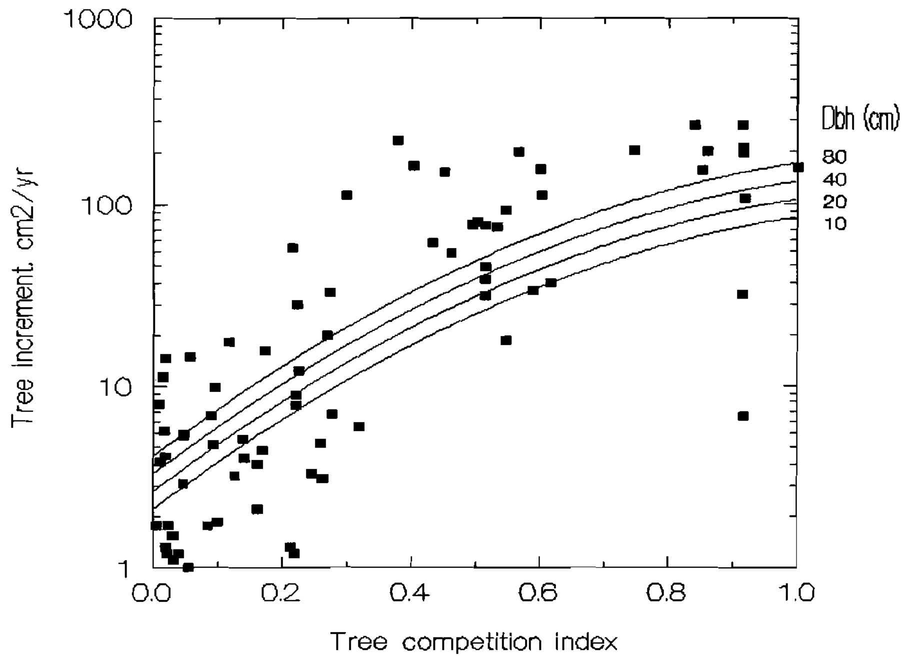


Figure 24 ***Graph of equation {50} with data for P.africanum for KADE plots. The regression relates increment to competition index and tree size***

Space

Site index, as applied in plantations and monospecific uniform stands (eg. Clutter et al. 1983:30-58; Alder, 1980:40-62), has very limited application. Canonizado (1979) used total height on the residual stand after logging as a site indicator for Phillipine dipterocarp forests under a uniform system of management. Generally, however, mixed tropical forests under selection management will not show any clear time-series of height development, nor may it be certain which species the site index will relate to or how it may be effected by logging of large trees.

Vanclay & Henry (1988) introduced the notion of site form, or mean tree height at an index diameter. This can be measured quite simply, although total tree height in MTF is difficult to assess. Bole height is however, not difficult to measure and is closely correlated with total height. If a sample is taken, as suggested in Alder & Synnott (1992) of height of the 2 largest diameter trees on each 20x20 m quadrat on PSPs (50 trees/ha), then it will be possible to build up height-diameter curves for common species by localities, and from this to assess either a single species-weighted site form for the stand, or separate site form parameter for each species.

The result will be a continuous variate, SF (site form) which will be a plot attribute. It can be used in multiple regressions directly, to build equations that combine site form with tree size and competitive status. The advantage of this approach is that the necessary information can be readily acquired during forest inventory operations. The disadvantage at this time is there is no certainty that site form will be usefully correlated with growth rate, as the application of this method was to a relatively simple type of mixed forest dominated by *Callitris*. It remains a method which should be tested on a moderate scale before being assumed to be generally applicable.

Another approach to site classification in MTF using continuous variates requires PSP data. This involves assessing the increment of a component of the stand and comparing it with overall increment levels. This can be used to derive a Growth Index (GI), suitably scaled relative to mean growth rates. The disadvantage of this technique is that it requires, *a priori*, measures of increment and therefore cannot be applied to temporary inventory plots. Vanclay (1989a) showed however that a growth index was nonetheless useful in screening possible static predictors of increment.

The final method of incorporating site variation is crude, in terms of spatial resolution, but is yet considerably better than ignoring the problem entirely. This involves categorizing plots on some basis that appears to be significantly correlated with growth rates. Possible factors include:

- Soil type or soil parent material
- Rainfall, or rainfall during the 6 driest months
- Vegetation association or forest type
- Geographic regions or forest blocks

FRESV code	Location (Forest Reserve)	No. sample trees of Wawa
ASAT	Assin Attandaso FR	30
ASUK	Asukese FR	733
BOBR	Bobiri FR	222
EBNI	Esuboni FR	62
SUPG	Supong FR	106
TOTN	Tonton FR	91
WAWH	Wawahi FR	23
Total		1267

Table 7 *Forest reserve codes and number of sample trees of Wawa in site analysis example*

This approach is illustrated with some data for growth rates of *Triplochiton scleroxylon* (Wawa) taken from the PSP database discussed on page 43 from Ghana. The data is classified by the forest reserve code in a XBASE variable called FRESV. The possible values of FRESV and the sample size for each reserve are shown in Table 7.

The data was imported into a SYSTAT file and the regression variables LBAI and LTBA constructed in order to fit the increment equation {41}. The FRESV code in SYSTAT is a categorical variable denoted by a $ suffix, FRESV$. Equation was modified to include FRESV$ as follows:

$$\log(BAI) = \alpha + b.FRESV\$ + c.\log(TBA) + d.TBA \qquad \{51\}$$

The categorical quantity, FRESV$ is coded in the statistical analysis by automatically generating one column (regression variable) for each of the possible values ASAT to TOTN. The last category, WAWH does not need to be coded as it is implicit in the constant term of the regression; it may be called the *base category*. The codes employed depend on the type of model, but for DUMMY coding, as suggested, these artificial variables will be zero for trees outside a specified reserve, and 1 for those within it. The user does not need to be concerned about this, as it is done automatically, but it is useful to understand the principles involved.

When the regression is fitted, the output is as shown in Example 48. The commands required are shown at the top of the example box. In SYSTAT, in addition to the regression commands, CATEGORY declares that FRESV$ is categoric and defines the type of coding. PRINT=LONG ensures a full printout, otherwise information on the coefficient values is suppressed in categoric models, which are assumed to be purely for analysis of variance purposes.

The program lists the categories encountered, in either numeric or alphabetical order. The last category listed is always the base category. Then, instead of a list of coefficient values, a list of 'effects' is printed. With DUMMY coding selected, these can be translated directly into coefficients. The table below shows the respective coefficient values derived from the output in Example 48 for the increment equation for each reserve.

Site	*a*	b	c	d
ASAT	-0.151	0.063	0.568	-0.000103
ASUK	-0.151	-0.239	0.568	-0.000103
BOBR	-0.151	0.407	0.568	-0.000103
EBNI	-0.151	0.319	0.568	-0.000103
SUPG	-0.151	0.033	0.568	-0.000103
TOTN	-0.151	-0.254	0.568	-0.000103
WAWI	-0.151	0.0	0.568	-0.000103

```
Commands use to run the regression:

        MODEL LBAI = CONSTANT+FRESV$+LTBA+TBA
        CATEGORY FRESV$/DUMMY
        ESTIMATE

Printed output:

 LEVELS ENCOUNTERED DURING PROCESSING ARE:
 FRESV$
  ASAT          ASUK          BOBR          EBNI          SUPG          TOTN
  WAWH
 ----------------------------------------------------------------------------
     5 CASES DELETED DUE TO MISSING DATA.

 DEP VAR:    LBAI      N:    1262  MULTIPLE R: 0.637  SQUARED MULTIPLE R: 0.405
                                -1
 ESTIMATES OF EFFECTS  B = (X'X)  X'Y

                                LBAI
   CONSTANT                     -0.151
      FRESV$  ASAT               0.063
      FRESV$  ASUK              -0.239
      FRESV$  BOBR               0.407
      FRESV$  EBNI               0.319
      FRESV$  SUPG               0.033
      FRESV$  TOTN              -0.254
       LTBA                      0.568
        TBA                 -.103118E-03

                      ANALYSIS OF VARIANCE

 SOURCE        SUM-OF-SQUARES   DF   MEAN-SQUARE     F-RATIO        P

 FRESV$              81.640     6       13.607      13.220 .146549E-13
 LTBA               412.530     1      412.530     400.818 .999201E-15
 TBA                 13.886     1       13.886      13.492 .249689E-03

 ERROR             1289.615  1253        1.029
```

Example 48 *Output from regression of increment on categorical site factors and tree size*

Figure 25 shows the regressions on linear axes, together with the original data. It will be noted that the R^2 of the fitted model in Example 48 is only 40.5%. Thus some 60% of the variation in log(increment) is unaccounted for by the regression. This low degree of fit is quite typical of increment regressions, which will usually have R^2s ranging from 0.2 to 0.7 depending on species and the quality of the data.

It can be seen that the reserves appear to fall into 3 groups. Bobiri (BOBR) and Esuboni (EBNI) are clearly good sites for Wawa; Assin Attandaso (ASAT), Supong (SUPG) and Wawahi (WAWH) are very similar and intermediate; whilst Tonton (TOTN) and Asukese (ASUK) show the lowest growth rates. The significance of the difference in growth rates between the reserves can be tested relatively easily in most statistics packages which offer a general linear modelling facility.

In SYSTAT, the method involves first of all fitting the regression, and then using the HYPOTHESIS command. This introduces various possible subcommands that establish a null hypothesis to be tested. For example:

```
HYPOTHESIS
EFFECT=FRESV$
SPECIFY
FRESV$['SUPG']=0.5*FRESV$['BOBR']+0.5*FRESV$['EBNI']
TEST
```

tests the null hypothesis that the effect of SUPG is the same as the mean effect of BOBR and EBNI. The ANOVA resulting from this shows a probability of 0.007 that the null hypothesis is correct, leading it to be rejected. In this way one can determine that ASUK and TOTN are one group, ASAT, SUPG and WAWH another, and BOBR, and EBNI a third. It would be possible to recode them for example, as site

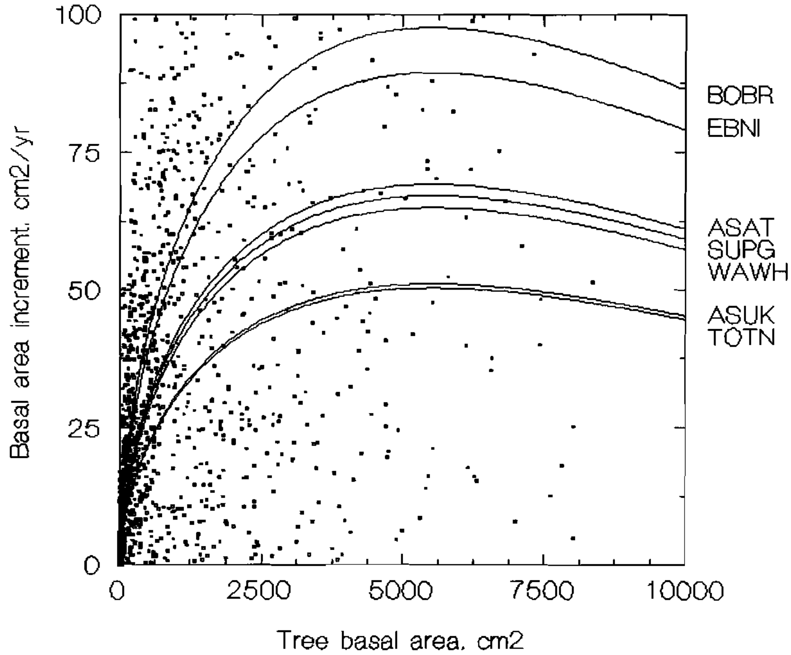


Figure 25 ***Triplochiton scleroxylon increment data from 7 forest reserves in Ghana, and fitted log-exponential model with a site-dependent categorical term***

types I, II, and III on a new categoric variable to be called SQ$. In the DATA module of SYSTAT, for example, this would be done with the commands:

```
LET SQ$=FRESV$
CODE SQ$ / 'BOBR'='I','EBNI'='I','ASAT'='II','SUPG'='II',
        'WAWH'='II','ASUK'='III','TOTN'='III'
RUN
```

The regressions could then be re-analysed using the site quality codes to determine mean lines for each of the 3 groups of reserves.

It will be clear that to undertake these kinds of analyses with many sites and species is tedious and time consuming. However, once the basic concepts have been tested and the principles of the analysis worked out, the various intermediate steps can be partially automated in one of two ways:

- Specialized programs can be written that use published multiple regression subroutines (eg. Press et. al., 1989).
- SYSTAT and other major statistical packages have Macro languages that allow repetitive analyses to be coded as procedures.

4.2.7 *Increment distributions and stochastic increment functions*

Stochastic models of increment seek to predict the parameters of the increment probability distribution. They contain more information than a deterministic model, which predicts solely the mean, and can potentially offer a better modelling strategy, especially given the high degree of variability for increment and the low coefficients of determination (R^2) obtained for deterministic regressions.

To fit a stochastic model, two stages are necessary:

- A suitable probability distribution must be be determined. As will be seen, the Weibull distribution appears to fit diameter or basal area increment well, and is an easy distribution to fit and use in simulation (Bailey & Dell, 1973).
- The parameters of the distribution are then related to factors such as tree size and competitive status to give predictive regressions for the distribution.

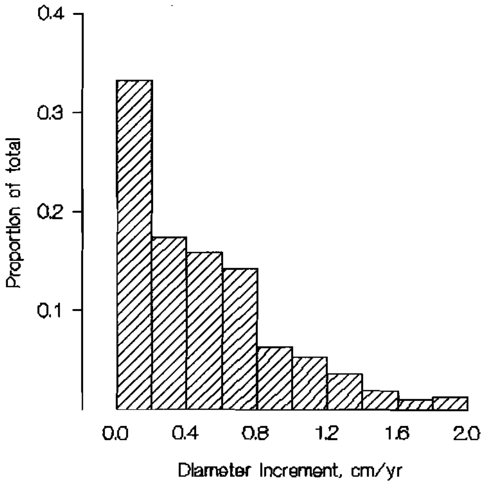


Figure 26 *Frequency distribution of Khaya ivorensis increments*

Figure 26 shows the frequency distribution for the *Khaya ivorensis* increment data shown in Figure 16. It can be seen that it is markedly skewed and not at all normal. The distribution of basal area increment is very similar, with an even greater concentration at the lower values.

When this data is plotted using classes with $\log_2$ intervals (such that the lower and upper bounds of each class are twice the previous class), then a different pattern is seen, more symmetric, but still over-dispersed compared to the normal distribution (Figure 27). The same effect can also be achieved by plotting the log of increments using fixed class intervals. This transformation shows that log(increment) is a more suitable variate for regression and ANOVA studies than raw diameter or basal area increment. However, the distribution is still not completely normal, and it is not a sufficiently accurate description of the distribution for stochastic modelling purposes.

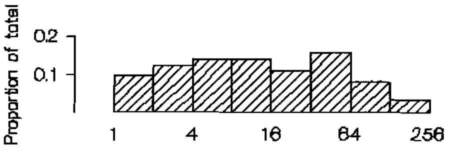


Figure 27 *Frequency distribution of K. ivorensis basal area increments using $\log_2$ class intervals*

The Weibull distribution (Bailey & Dell, 1973) appears to give a good empirical fit to the frequency distribution of diameter increment, but appears to be less effective as a description for basal area increments. The Weibull function is most usefully expressed in its cumulative form as:

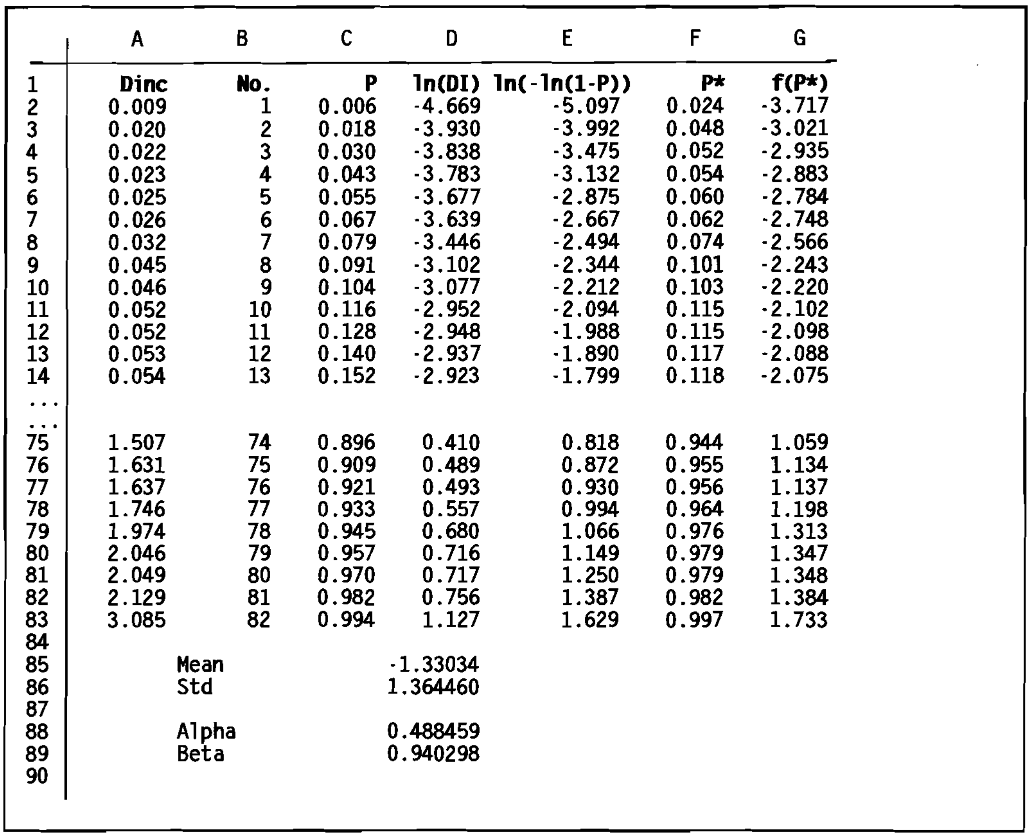
$$P = 1 - \exp(-(x/\alpha)^{ß}) \qquad \{52\}$$

where P is the probability that a value of ≤x will occur. The α coefficient is the *scale* parameter; its value corresponds to the 63.2% point on the cumulative distribution. This implies that 63.2% of all values of x from the distribution will be less than α. The ß coefficient is the *shape* parameter, governing the appearance of the probability density function or frequency histogram. Figure 12 on page 68 shows some examples of shape with different values of ß.

In SYSTAT, increments can be plotted on Weibull probability axes directly using the PPLOT command with the WEIBULL option. However, one can gain more insight into the procedures involved by doing the necessary calculations on a spreadsheet. Example 49 shows a set of data for *Piptadeniastrum africanum* diameter increments from Kade, Ghana (Swaine et al., 1987). It shows how the data may be organized to produce the Weibull probability plot shown in Figure 28, and to estimate Weibull parameters. The procedures involved in building the spreadsheet are as follows:

- The increment data is imported, as a single column of numbers, into column A. This may be in random order.

- The data is sorted using /Data Sort (in Lotus 123) into ascending order.



	A	B	C	D	E	F	G
1	**Dinc**	**No.**	**P**	**ln(DI)**	**ln(-ln(1-P))**	**P***	**f(P*)**
2	0.009	1	0.006	-4.669	-5.097	0.024	-3.717
3	0.020	2	0.018	-3.930	-3.992	0.048	-3.021
4	0.022	3	0.030	-3.838	-3.475	0.052	-2.935
5	0.023	4	0.043	-3.783	-3.132	0.054	-2.883
6	0.025	5	0.055	-3.677	-2.875	0.060	-2.784
7	0.026	6	0.067	-3.639	-2.667	0.062	-2.748
8	0.032	7	0.079	-3.446	-2.494	0.074	-2.566
9	0.045	8	0.091	-3.102	-2.344	0.101	-2.243
10	0.046	9	0.104	-3.077	-2.212	0.103	-2.220
11	0.052	10	0.116	-2.952	-2.094	0.115	-2.102
12	0.052	11	0.128	-2.948	-1.988	0.115	-2.098
13	0.053	12	0.140	-2.937	-1.890	0.117	-2.088
14	0.054	13	0.152	-2.923	-1.799	0.118	-2.075
...							
...							
75	1.507	74	0.896	0.410	0.818	0.944	1.059
76	1.631	75	0.909	0.489	0.872	0.955	1.134
77	1.637	76	0.921	0.493	0.930	0.956	1.137
78	1.746	77	0.933	0.557	0.994	0.964	1.198
79	1.974	78	0.945	0.680	1.066	0.976	1.313
80	2.046	79	0.957	0.716	1.149	0.979	1.347
81	2.049	80	0.970	0.717	1.250	0.979	1.348
82	2.129	81	0.982	0.756	1.387	0.982	1.384
83	3.085	82	0.994	1.127	1.629	0.997	1.733
84							
85		Mean		-1.33034			
86		Std		1.364460			
87							
88		Alpha		0.488459			
89		Beta		0.940298			
90							

Example 49 *Spreadsheet of diameter increment data organized for graph of Weibull order statistics and calculation of Weibull parameters*

- Column B is built up giving the order statistic of each value. The first cell, B2 is set to 1. Cell B3 is set to 1+B2. Cell B3 is then copied down using /Copy for the remaining data, giving a sequential set of numbers.

- Column C is the cumulative probability estimated for each order number, using Yates' correction for continuity, given as:

  $$P = (i - ½)/n \qquad \{53\}$$

  where i is the order number, and n is the total number of increments, given by the value of cell B83. The Lotus formula for calculating the cells of column C is:

  (B2-0.5)/$B$83

  This is again copied down from C2 to the remaining cells.

The Weibull function in equation {52} can be linearized in the parameters by noting that:

$$\log(-\log(1-P)) = k + \beta.\log(x) \qquad \{54\}$$

where k is $-\alpha^{\beta}$. Consequently plotting log(-log(1-P)) against log(DI) should give a straight line if the data is indeed distributed according to the Weibull function. The Weibull parameters can be estimated from equation {54} by regression, but this is unnecessarily complicated. Nelson (1982) gives a simple method that does not require order statistics. Given the natural logs of a variate x (such as increment) as:

$$y = \log(x)$$

Then:

$$\text{ß} = 1.283 / \sigma_y \qquad \{55\}$$

and $\alpha = \exp(\mu_y + 0.5772/\text{ß})$ {56}

This method is said to be unbiased, although less efficient* than methods based on weighted regressions using the order statistics such as D'Agostino (1971). It can be applied only to uncensored, ungrouped, data.

For the spreadsheet, the following further stages are then involved:

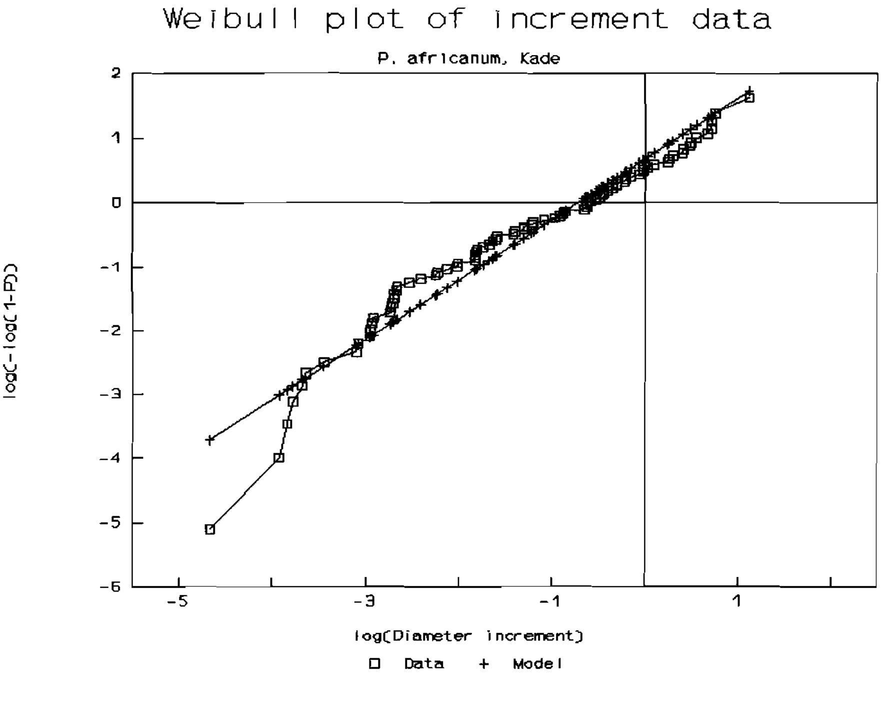


Figure 28 ***Weibull probability plot constructed using Lotus 123. The data are <u>Piptadeniastrum africanum</u> diameter increments from University of Ghana plots at Kade***

* That is, the estimators from this method have higher variance than for some other techniques.

- Cell D2 is calculated as @LN(C2) and copied down for the remaining data.

- The mean and standard deviation of this column are calculated in cells D85 and D86 using the following Lotus functions:

  D85 = @AVG(D2..D83)
  D86 = @STD(D2..D83)

- The Weibull parameters α and ß are calculated in cells D88 and D89 using equations {55} and {56}. The formulae are:

  D89 = 1.283/D86
  D88 = @EXP(D85 + 0.5772/D89)

- The transformed probabilities for each order statistic are calculated in column E. Cell E2 is defined as

  E2 = @LN(-@LN(1-C2))

  This is copied down for the remaining cells in the column.

- At this stage, the graph of data can be examined using the /Graph command, selecting type XY, setting the X range to D2..D3 and the A range to E2..E83.

- The predicted Weibull distribution can be plotted by calculating the values in column F. These are the probabilities associated with each diameter increment predicted by the model. The first cell is given by:

  F2 = 1-@EXP(-(A2/$D$88)^$D$89

  This is simply the Weibull function {52}, with α as cell D88 and ß as cell D89. The $ operators denote that these cell references are not altered during copying operations. Cell F2 is copied down for the remainder of the data.

- Column G is necessary to graph the predicted probabilities, and is simply the transformation of column F:

  G2 = @LN(-@LN(1-F2))

  copied down for the remining cells. The range G2..G83 can be set as data range B in the graph, and plotted to produce the result shown in Figure 28.

Although these explanations are somewhat detailed it will be found that the procedures involved are very simple. As noted previously, SYSTAT can directly plot cumulative probability graphs on Weibull axes. There is however an error in the SYSTAT documentation (upto version 5.0) regarding the nature of the scales on these graphs. The abscissa (y axis) is log(-log(1-P)), and not log(P) as stated.

Once the suitability of a stochastic model such as the Weibull function has been explored, the next step is to develop predictive regressions for the distribution parameters. This requires that a file is created with the distribution parameters as dependent regression variables, and tree parameters such as species, diameter and competitive status as predictor variables. To build such a file involves fitting large numbers of distributions to sets of increments from trees classified by the selected predictor variables.

The process is illustrated here for a model in which Weibull coefficients of diameter increment distributions are to be estimated from species and crown class. The method can be adapted as required by substituting diameter class or some other tree competition index for crown class.

The data classification and calculation of the Weibull coefficients can be done entirely in XBASE using interactive commands given from the dot prompt. Nelson's method for estimating Weibull α and β is used (see equations {55} and {56} on page 129). The illustration is based on the data file of tree increments from old-series PSPs in Ghana discussed in Example 19 on page 43.

The sequence of operations performed from the dot prompt can be discussed in terms of the principle commands in the order that they are used.

- CREATE DIWFN

  The process starts with the creation of a file having the structure shown in Example 50. The fields SPP, CPOS and DI must correspond in name and type to the fields in the original PSP increment database. This file is called here DIWFN (Diameter Increment Weibull Function)

- USE DIWFN
  APPEND FROM PSPINC FOR DI>0

  Data fields in the main data file are appended. Only those fields with coincident names will be copied, which in this case will be SPP, CPOS and DI. Trees with zero or negative increments are excluded from the analysis. This will exclude both dead trees, and some live trees with negative increments. The question of how the latter may be dealt with within a stochastic model is discussed later.

```
Field  Field Name  Type       Width   Dec   Index
    1  SPP         Character     3              N
    2  CPOS        Numeric       1              N
    3  SPPXCPOS    Character     4              Y
    4  DI          Numeric       5      2       N
    5  N           Numeric       5              N
    6  LDI         Numeric       9      5       N
    7  LDISQ       Numeric      14      8       N
```

Example 50 *Structure of DIWFN database used as intermediate step in estimating Weibull coefficients by species and crown classes*

-
  ```
  REPLACE ALL LDI WITH LOG(DI), LDISQ WITH LDI*LDI, ;
          SPPXCPOS WITH SPP+STR(CPOS,1), N WITH 1
  ```

  The above command fills in the fields SPPXCPOS, LDI and LDISQ. SPPXCPOS is used to control the subsequent TOTAL operation, and combines the separate SPP and CPOS fields into a single entity. LDI contains log(DI), and LDISQ is filled with LDI^2. The field N is set to 1 for every record. After the TOTAL operation, it will contain the count of the number of cases in each SPP x CPOS cell.

-
  ```
  INDEX ON SPPXCPOS TAG SPPXCPOS
  ```

  The file is indexed on the combined species, crown position values in SPPXCPOS. This is required for the TOTAL operation to work correctly.

-
  ```
  TOTAL ON SPPXCPOS TO DIWFNT FIELDS DI, N, LDI, LDISQ
  ```

  A new file is created called DIWFNT with the same field designators as DIWFN. Fields DI, N, LDI and LDISQ will be the totals for each value of SPPXCPOS of the equivalent fields in DIWFN. In other words, LDI in the output file is the sum of the log(diameter increments), LDISQ is the sum of squares, N is the count of cases, and DI will be the total of diameter increments.

-
  ```
  USE DIWFNT
  MODIFY STRUCTURE
  ```

  The output file from the TOTAL is made the active file, and its structure is modified interactively to add fields called MU, SIGMA, ALPHA and BETA. These will contain the values μ, σ, α and β used in equations {55} and {56}. The modified structure will appear as shown in Example 51.

-
  ```
  REPLACE ALL MU WITH LDI/N, ;
          SIGMA WITH SQRT((LDISQ-LDI*LDI/N)/(N-1)), ;
          BETA WITH 1.283/SIGMA, ;
          ALPHA WITH EXP(MU+0.5772/BETA)
  ```

  This performs the calculations of equations {55} and {56}. MU is the mean of the log(increments), SIGMA is the standard deviation. The Weibull coefficients ALPHA and BETA for the distribution of increment are estimated by Nelson's method.

-
  ```
  COPY TO DIWFNX FIELDS SPP,CPOS,DI,N,ALPHA,BETA
  USE DIWFNX
  REPLACE ALL DI WITH DI/N
  ```

  This prepares the file for export to SYSTAT by copying only the fields needed for regression analysis. The DI field is also converted at this stage from a total to a mean value. Neither DI nor N are directly required, but

```
Field  Field Name  Type       Width    Dec    Index
  1    SPP         Character    3               N
  2    CPOS        Numeric      1               N
  3    SPPXCPOS    Character    4               N
  4    DI          Numeric      5       2       N
  5    N           Numeric      5               N
  6    LDI         Numeric      9       5       N
  7    LDISQ       Numeric     14       8       N
  8    MU          Numeric      7       4       N
  9    SIGMA       Numeric      7       4       N
 10    ALPHA       Numeric      7       4       N
 11    BETA        Numeric      7       4       N

SPP CPOS SPPXCPOS   DI     N      LDI          LDISQ      MU    SIGMA  ALPHA   BETA
Ant    1 Ant1      5.98   33  -72.10374  187.57268778 -2.1850 0.9687 0.1739 1.3245
Ant    2 Ant2     42.46  176 -338.5373   849.10382918 -1.9235 1.0635 0.2357 1.2064
Ant    3 Ant3     62.43  160 -238.3527   554.21466140 -1.4897 1.1191 0.3730 1.1465
Ant    4 Ant4     85.91  100  -76.31612  185.15768342 -0.7632 1.1322 0.7758 1.1332
Ant    5 Ant5     33.84   44  -29.39694   74.56650658 -0.6681 1.1302 0.8524 1.1352
Chl    2 Chl2      6.57   27  -50.74785  124.14909657 -1.8796 1.0518 0.2450 1.2198
Chl    3 Chl3     21.74   52  -66.49150  131.57535268 -1.2787 0.9554 0.4279 1.3429
Chl    4 Chl4     14.62   33  -42.14237   94.51059396 -1.2770 1.1277 0.4632 1.1377
Chl    5 Chl5      8.08   12  -12.91077   35.73608407 -1.0759 1.4092 0.6428 0.9104
Ea     1 Ea 1     12.06   79 -176.8070   456.66229408 -2.2381 0.8840 0.1588 1.4514
```

Example 51 *Structure of the database DIWFNT and a listing of the first 10 records. The calculated Weibull coefficients are shown as ALPHA and BETA. The DI field is the total of diameter increments for each SPPxCPOS cell*

may be useful for documenting the dataset.

Once the file has been imported in the statistical package, then the regressions of the Weibull coefficients can be examined species by species, and individually fitted. It will probably be found the α parameter closely correlates with tree size, whilst the ß shows only a weak relationship. There will be a tendency for ß to increase from values around or less than 1 for the smallest understorey components to higher values for dominants and emergents.

However, a useful simplification arises if the relationship between α and the mean diameter increment is plotted and analysed. Figure 29 shows the result for the DIWFNX data derived from PSPs in Ghana. The slope of the regression line is 1.002 and the intercept 0.007. Using the HYPOTHESIS command in SYSTAT, it is found that the slope does not significantly differ from 1, nor the intercept from zero. For this data set, the Weibull α is the same as the mean diameter increment for all species pooled.

Likewise ß is found to be essentially invariate, and can be treated as constant at about 1.25. The best fit depends on how outliers are treated and may range from 1.24 to 1.26.

One thus arrives in this case at a stochastic model that describes the distribution of increment, but which is essentially compatible with and derived from a model to predict mean diameter increment.

The reader should however test this approach carefully for particular data sets. The α coefficient will always closely relate to mean diameter as it represents the 63% quantile of the frequency distribution. However, ß is unlikely to appear

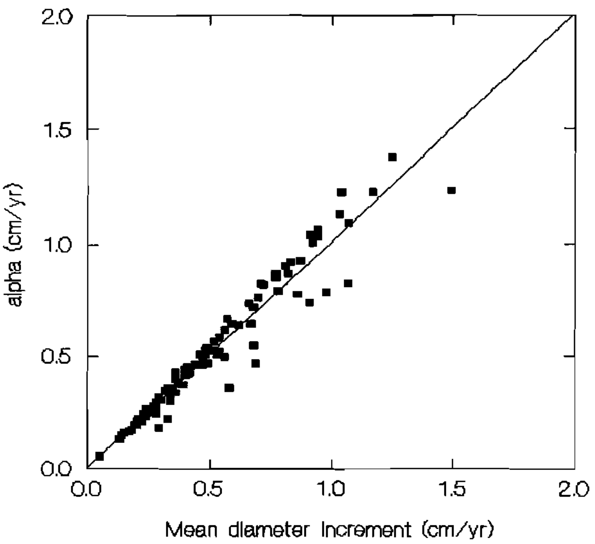


Figure 29 *The regression of Weibull α on mean diameter increment shows a slope close to unity and zero intercept, using data from PSPs in Ghana (all species combined)*

invariate if it is well defined by plentiful data over a range of sizes.

4.3 Mortality functions

4.3.1 *Causal factors in tree mortality*

Mortality is the rate at which trees die, and is generally expressed as a percentage per year. Vanclay (1994:173) distinguishes between regular and catastrophic mortality. Regular mortality can be assigned to predictable causes such as stand density, tree size, logging damage, and so on; catastrophic mortality is generally associated with abnormal and infrequent events such as hurricanes or rare intense fires.

Regular mortality is predictable over long periods of time, but may be episodic in any one year. There is no precise differentiation between regular and catastrophic mortality. Moderately severe storms may, for example, trigger tree falls or breakage on stems weakened by internal decay, resulting in a synchronised high mortality in one year, with several subsequent years of lower mortality.

Regular mortality can be attributed to the following main factors (Oliver & Larson, 1990):

- *Shading:* Depending on the ecology of a given species, it will be more or less able to tolerate long periods in deep shade. Strongly light-demanding species can be expected to show higher mortality as understorey trees than those which are shade tolerant.

- *Fungal infection:* Rot in the stems or main branches greatly increases the probability that the tree will break during wind storms. Propensity to decay is associated with tree age or size, species, and factors that cause damage to the stem earlier in life, such as fires or moderate logging damage.

- *Insects, viruses, etc.:* The normal disease mechanisms that affect any organism can also kill forest trees, or may provide pathways for the entry of decay.

- *Logging damage:* Moderate logging damage will lead to fungal infection and breakage after some decades. Experimental evidence also shows that high mortality occurs on damaged stems over periods of less than a decade (Tang, 1976). Small trees, saplings and seedlings will be directly crushed and destroyed in great numbers during logging. Large trees may suffer massive crown or bark loss, or be pushed, weakening their root system and mechanical stability.

- *Exposure:* Trees which have become adapted to deep shade conditions often cannot withstand sudden exposure to intense sunlight as a result of natural gap formation or logging. This is again a species-related characteristic.

Mortality caused by shading is most likely to affect small trees. Fungal infection, leading to wind falls or breakage, is typically associated with large, old trees, but may occur at any age as a result of insect infestations or physical damage. Death from sudden exposure to light is also associated with small, shade-adapted trees. Logging damage affects all sizes but tends to cause abrupt mortality in smaller trees, and have more lingering consequences among the larger ones.

The coded notes suggested in Alder & Synnott (1992) for use on PSPs are intended to assist in identifying these causal factors and provide better quantitative information about them than is currently available. Most older series PSPs have measurement protocols that assume increment to be the most important factor in stand dynamics, with mortality often being observed only by default, ie. by the absence of a tree after a sequence of measurements. However, simulation models of stand growth show that the structure of the stand table and the ultimate timber yield is at least as sensitive to mortality as it is to mean tree increment, and probably more so. Accurate observations on tree mortality and any identifiable predisposing factors are therefore most important on PSPs.

4.3.2 *Typical mortality rates*

Mortality rates over a range of sizes in mixed tropical forest have typically be found to vary from 1% to 5%. Table 9 shows some values that have been noted by different authors. Considering all species and trees of 10 cm dbh or larger, typical mortality rates are of the order of 1.5% per annum in forest that has not been recently disturbed. However, in forests where logging or other disturbances have occurred in the previous decade, higher mortality rates, of the order of 2.5% or more, may be expected.

Citation	Forest type & location	Mortality rates, %/year		Comments
		Poles ≤ 10 cm dbh	Larger trees	
Alder (1993)	Semi-deciduous MTF, Bobiri, Ghana	(All sizes combined) Shade Bearers: 0.3-0.7% Non-Pioneer Light Demanders: 1.2-1.5% Pioneers: 3-4.5%		Mortality rates over 15 year period depend greatly on species and ecological guild. Forest heavily logged 10-15 yrs prior to start of measurements.
Bryan (1980)	Dipterocarp, Sarawak	4-5%	2-3%	PSPs established less than 10 years after logging, measured over 5 years
Nicholson (1979)	Dipterocarp, Sabah		1.06%	Unlogged forest
Mervart (1972)	Nigeria, humid mixed forest	5.5%	3.5%	6,224 trees on PSPs observed over 5 years. Difference between size classes not significant.
Singh (1981)	India, Wet evergreen forest		1.11%	Significant differences between species, but no correlation with growth rate
Swaine (1992)	Semi-deciduous MTF, Kade, Ghana	3.4-6.5%	1.5%	Unlogged forest measured over 19 years. Higher mortality on suppressed trees.
Tang (1976)	Trengganu, W. Malaysia, Dipterocarp forest		2.8-10.2%	5-year period following logging. Mortality highly correlated (R^2 0.62) with % basal area removed.
Silva (1989)	*Terra firme* forest, Tapajos, Brazilian Amazon	3-4%	1.5-2.5%	No correlation of mortality with increment. Logged forest, with rates 1-2% higher in period immediately after logging
British Honduras Forestry Dept. (1949)	Mixed broadleaf forest, Belize		2.7%	Swietenia macrophylla over 8 year period, logged & hurricane disturbed forest.

Table 9 *Mortality rates noted in mixed tropical forest studies by various authors*

Mortality rates, like growth rates, depend on species, tree competitive status, and on site conditions. Among small trees ($\leq$10 cm dbh), as may be expected, mortality under shade conditions tends to be associated with ecological guild. Pioneers may show high mortality, of the order of 5%/yr or more, whilst shade bearers may have very low mortality rates, less than 1%.

For any given species, the mortality rate by size class tends to show only weakly defined differences above 10cm dbh, but with increasing mortality below that size, typically of the order of 3-5%/yr. There may be a tendency for mortality to rise again above average levels for the largest trees, associated with their greater propensity to decay and their exposure to greater wind stresses in an emergent position above the canopy (Alder, 1990).

Some authors have noted that mortality tends to be associated with trees whose increment is neglible. In some models of temperate mixed forests, this has been the basis for assigning mortality probability to a tree (eg. Buchman, 1979). However, this is not invariably the case, and only some part of regular mortality (that directly attributable to suppression) is likely to be linked to low increment. Alder (1993) found that actively growing, apparently health canopy trees would die from wind damage. In other cases, actively growing trees would show negative growth immediately prior to death, probably due either to exfoliation of the bark, or actual wood shrinkage associated with internal drying of conductive tissue. The causal factors for these large, apparently healthy trees was not indicated in the data, but could have included fire damage, insect attack, viral diseases, or root fungi among other possibilities.

4.3.3 *Calculation methods for deriving mortality rates from PSP data*

Dead trees may be indicated in a sequence of measurements by a variety of methods. Most commonly, a field of coded notes, such as the NOTES field in Example 2 on page 12 will contain codes to indicate that a tree is dead. The XTAB cross-tabulation program discussed in section 2.5.7 (page 44 ff.) shows how dead trees may be counted by diameter classes. Appropriate modifications to this program can allow numbers of dead trees to be counted by competition classes, sites, or other factors, as well as diameter classes.

Given an output data file from the cross-tabulation similar to that shown in Example 21, then the periodic mortality rate PMORT can be calculated as:

$$\text{PMORT} = \text{DEAD/TREES}$$

where DEAD is the number of dead trees at the end of a period in the class, and TREES is the total number of sample trees, both dead and alive. If the measurement period is MYR years, then the annual mortality rate AMORT can be derived from:

$$\text{AMORT} = 1 - (1\text{-PMORT})^{1/\text{MYR}} \qquad \{57\}$$

Equation {57} is derived by considering that survival after one year, with annual mortality AMORT, will be:

$$SURV = 1 - AMORT$$

In other words, if annual mortality is 1%, survival at the end of the year will be 99% of the initial population. After 2 years, survival would be:

$$SURV = (1 - AMORT)^2$$

and after MYR years it will be:

$$SURV = (1 - AMORT)^{MYR}$$

Substituting the periodic mortality for periodic survival one has:

$$(1 - PMORT) = (1 - AMORT)^{MYR}$$

which can be transposed to give equation {57}

Unfortunately, in many cases the measurement intervals for PSPs may be quite unequal, partly as a result of administrative factors that result in missed or late measurements on plots, and partly because often data sets from different sources may be combined for analysis, such as experimental plots and passive monitoring PSPs.

In such cases, the data should be indexed on the measurement interval, and mortality rates calculated for each set of periods separately. These may then be converted to annual rates, using equation {57} and an overall average calculated weighted by the number of trees sampled over each measurement interval.

This approach may be illustrated using some data from old-series PSPs in Ghana (Alder, 1990). The GHINC data file shown in Example 52, compiled from raw data in text form, gives one record per tree, with species, diameter (cm), increment (cm/yr), and crown position among other factors. Dead trees are indicated by a value of -99 in the DI (diameter increment) field. The measurement interval is shown in the YINT field in years, and ranged in value from 3 to 11 years for different plots.

In order to use the XBASE TOTAL command to compile mortality counts by species and crown position, the structure of GHINC was modified in a second, temporary, file called MORT1. The following fields were added:

MIX This was used to provide a key for the TOTAL command, and was defined as a 6-character field, to be filled with the concatenated values of YINT, SPP and CPOS. The database was indexed on MIX after it had been created. The defining DBASE commands were:

Structure of the GHINC database

```
Field  Field Name  Type       Width   Dec  Content

    1  FRESV       Character    4          Forest reserve code
    2  PNUM        Numeric      3          Plot number
    3  LDNO        Numeric      3          Tree (Leading Desirable) No.
    4  SPP         Character    3          Species code
    5  DBH         Numeric      6      1   DBH, cm
    6  DI          Numeric      5      2   Diameter increment, cm/year
    7  CPOS        Numeric      1          Dawkins crown position score
    8  NCT         Numeric      2          No. competing trees
    9  YINT        Numeric      2          Measurement interval, years
```

Listing of part of the database Note that dead trees are indicated by an increment of -99 in the DI field.

```
FRESV PNUM LDNO SPP   DBH    DI CPOS NCT YINT

WAWH     1   34 ANT   8.0 -99.0   3   5    5
WAWH     1   35 NES  37.1  0.24   3   8    5
WAWH     1   36 MAN  27.2  0.57   3   4    5
WAWH     1   37 TRI  34.3  0.79   3   7    5
WAWH     1   38 ANT  51.5  0.04   3   6    5
WAWH     1   39 GC   11.4  0.34   2   8    5
WAWH     1   40 GC   17.6  0.47   3   4    5
WAWH     1   41 GC    5.5 -99.0   2   8    5
WAWH     1   42 GC    8.5  0.20   2   6    5
WAWH     1   43 GC   10.2  0.36   2   8    5
WAWH     1   44 KI    5.8 -99.0   2   3    5
WAWH     1   45 ANT  47.0 -99.0   3   9    5
WAWH     1   46 NES  29.3  1.01   3   5    5
WAWH     1   47 P    16.7  0.06   3   7    5
WAWH     1   48 MAN   0.0 -99.0   1   5    5
WAWH     1   49 GC   11.6  0.00   3   4    5
```

Example 52 ***Record structure and partial listing of mortality data from Ghana old-series PSPs. This data is used to demonstrate mortality calculation over variable measurement periods.***

```
REPLACE ALL MIX WITH STR(YINT,2)+SPP+STR(CPOS,1)
INDEX ON MIX TAG MIX
```

NTREES This is used to count the number of trees (dead and alive) in each class formed by the various MIX values. It is set initially to 1, but will contain class counts after using TOTAL. It is filled with the following command:

```
REPLACE ALL NTREES WITH 1
```

NDEAD This is used to count dead trees in each MIX class. Prior to the TOTAL it is set for each tree record to zero for live trees, and 1 for dead trees, using the following command:

```
REPLACE ALL NDEAD WITH IIF(DI=-99,1,0)
```

The TOTAL command is given as follows:

```
TOTAL ON MIX TO MORT2 FIELDS NTREES,NDEAD
```

Only the fields YINT, SPP, CPOS, NTREES, NDEAD and MIX contain meaningful information in the output file MORT2. Fields DBH, DI, and NCT are not used for the analysis (which is solely by species and crown position) and should be ignored.

To proceed to derive annual mortalities, additional fields are added to the MORT2 file. These are:

MIX2 A second index field, for totalling on species and crown position over the various measurement periods. This is filled with:

```
REPLACE ALL MIX2 WITH SPP+STR(CPOS,1)
```

AMORT This is the annual mortality calculated using equation {57} with:

```
REPLACE ALL AMORT WITH 1-(1-NDEAD/NTREES)^(1/YINT)
```

WAM This will contain AMORT weighted by tree numbers, NTREES, and is used in calculating the mean value of AMORT for all the different measurement periods. It is calculated as:

```
REPLACE ALL WAM WITH AMORT*NTREES
```

The MORT2 file is placed in use as the active file, and then totalled on the key field MIX2 to derive totals of NTREES and weighted AMORT, WAM. The mean value of AMORT can than be calculated. These final steps are achieved with the following commands:

```
USE MORT2
TOTAL ON MIX2 TO MORT3 FIELDS NTREES,WAM
USE MORT3
REPLACE ALL AMORT WITH WAM/NTREES
```

The complete DBASE dialog is shown in Example 53. It will be noted that the separate REPLACE commands discussed above are combined. This is considerably faster when working with large data files.

The final stage of the process is to create a database file that retains only those fields in MORT3 that are meaningful, and possibly to delete the files MORT1, MORT2 and MORT3 once each stage of the analysis has been correctly performed. The final file shown, MORTSUM, contains only the species, crown class, total number of trees, and mean mortality.

The same procedure could be used to calculate mean mortalities by diameter class, by substituting a variable DCLASS for CPOS wherever the latter occurs in the example. DCLASS would be initially created as a field in MORT1, and filled with a command such as:

```
REPLACE ALL DCLASS WITH IIF(DBH<80,INT(DBH/10+1),9)
```

This would create 10-cm classes, with all trees greater than or equal to 80 cm in class 9 (see also the discussion on diameter class formation on page 42).

4.3.4 *Confidence limits for mortality rates*

Mortality data is distributed according to the Binomial distribution which governs sampling for data which may take one of two discrete values. In the case of mortality data, the values are dead or alive, which may be coded as 0 or 1 respectively.

The mean probability of mortality, p, is given by:

$$p = m/n \tag{58}$$

where:

m are the number of trees which die,
n are the total number of trees (both dead and alive).

The variance of the estimate of p is given by:

$$\sigma^2 = p.(1-p)/n \tag{59}$$

Confidence limits are asymmetric about p and derived from the F-distribution, using the following formulae (Nelson, 1982:205):

$$p_L = 1/[1+F(\alpha,2n-2m+2,2m).(n-m+1)/m] \tag{60}$$

$$p_U = 1/[1+(n-m)/\{(m+1).F(\alpha,2m+2,2n-2m)\}] \tag{61}$$

where:

n,m are total and dead trees respectively,
α is the confidence level (eg 95%),
p_L is the lower confidence limit for p,
p_U is the upper confidence limit for p,
F() is the F-distribution function.

Standard statistical tables frequently include charts of binomial confidence limits. These give approximate results and may be slow when dealing with any quantity of data. It is also possible to use approximations based the normal distribution when n and m are large (n,m $\geq$ 10); or on the Poisson distribution when m is small (m $\leq$ n/10).

The values of n, m and p will relate to periodic mortality. If it is desired to present periodic mortalities as annual rates, then the confidence limits should first be calculated as periodic values, and then each corrected to annual values using equation {57}.

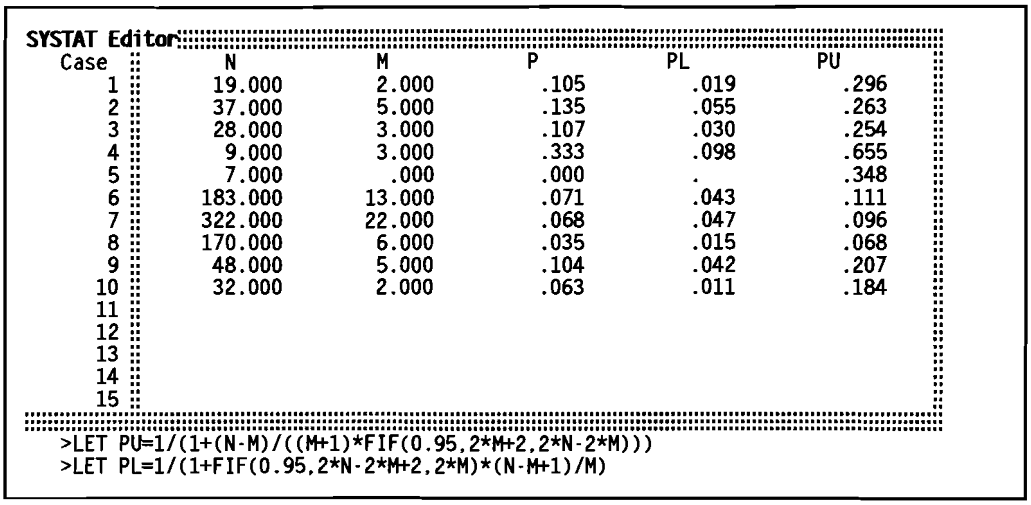
```
»USE GHINC
»COPY TO MORT1
  10817 records copied
»USE MORT1
»MODIFY STRUCTURE
  10817 records added
»REPLACE ALL MIX WITH STR(YINT,2)+SPP+STR(CPOS,1),NTREES WITH 1, NDEAD WITH IIF(DI=-99,1,0)
  10817 records replaced
»INDEX ON MIX TAG MIX
   100% indexed       10817 Records indexed
»LIST OFF NEXT 5

 FRESV PNUM LDNO SPP    DBH    DI CPOS NCT YINT MIX     NTREES  NDEAD
 WAWH     1   10 EC    16.8  0.55    3   7    5  5EC 3       1      0
 WAWH     1   21 EC    27.7  1.09    3   7    5  5EC 3       1      0
 WAWH     1   22 EC    19.6  0.22    3   7    5  5EC 3       1      0
 WAWH     1   50 EC    14.0  0.00    3   9    5  5EC 3       1      0
 WAWH     4   25 EC     5.3  0.12    3   4    5  5EC 3       1      0

»TOTAL ON MIX TO MORT2 FIELDS NTREES,NDEAD
  10817 records totaled
    437 records generated
»USE MORT2
»MODIFY STRUCTURE
    437 records added
»REPLACE ALL MIX2 WITH SPP+STR(CPOS,1),AMORT WITH 1-(1-NDEAD/NTREES)^(1/YINT),WAM WITH
AMORT*NTREES
    437 records replaced
»INDEX ON MIX2 TAG MIX2
   100% indexed         437 Records indexed
»LIST OFF NEXT 5

 FRESV PNUM LDNO SPP  DBH    DI CPOS NCT YINT MIX     NTREES  NDEAD MIX2    AMORT     WAM
 TOTN    17   40 ANT 14.6  0.15    1   8    4  4ANT1       1      0 ANT1  0.0000  0.0000
 SUPG    12   43 ANT  4.4  0.14    1   9    5  5ANT1       9      2 ANT1  0.0490  0.4410
 SUPG    12   43 ANT  5.2  0.34    1   8    6  6ANT1       5      1 ANT1  0.0365  0.1825
 ASUK    51   17 ANT  0.0 -99.0    1   8    9  9ANT1       3      1 ANT1  0.0441  0.1323
 ASUK    18   45 ANT  0.0  0.57    1   9   10 10ANT1       4      0 ANT1  0.0000  0.0000

»TOTAL ON MIX2 TO MORT3 FIELDS NTREES,WAM
    437 records totaled
    104 records generated
»USE MORT3
»REPLACE ALL AMORT WITH WAM/NTREES
    104 records replaced
»COPY TO MORTSUM FIELDS SPP,CPOS,NTREES,AMORT
»USE MORTSUM
»LIST OFF NEXT 5

 SPP CPOS NTREES    AMORT
 ANT    1     23   0.0329
 ANT    2    186   0.0293
 ANT    3    235   0.0168
 ANT    4     87   0.0166
 ANT    5     28   0.0156
```

Example 53 ***XBASE dialog to create mortality summaries from data measured over variable periods***

If measurements are made over variable periods (some plots 3 years, some 4, some 5 etc) then it will not be possible to calculate exact confidence limits. Approximate confidence limits for the pooled annual mortality can be shown graphically using a normal distribution approximation, provided that the number of dead trees is at least 10 for each species. Upper and lower confidence limits are calculated using equation {59} as:

$$p_{L|U} = p \pm t.\sigma \qquad \{62\}$$

where t is Student's *t*. The degrees of freedom to be used corresponds to the number of dead trees. The value of t may be taken as approximately 2 at a 95% confidence level with sufficient accuracy for the method.



```
SYSTAT Editor
 Case       N            M            P           PL           PU
    1     19.000       2.000        .105        .019         .296
    2     37.000       5.000        .135        .055         .263
    3     28.000       3.000        .107        .030         .254
    4      9.000       3.000        .333        .098         .655
    5      7.000        .000        .000         .           .348
    6    183.000      13.000        .071        .043         .111
    7    322.000      22.000        .068        .047         .096
    8    170.000       6.000        .035        .015         .068
    9     48.000       5.000        .104        .042         .207
   10     32.000       2.000        .063        .011         .184
   11
   12
   13
   14
   15
   >LET PU=1/(1+(N-M)/((M+1)*FIF(0.95,2*M+2,2*N-2*M)))
   >LET PL=1/(1+FIF(0.95,2*N-2*M+2,2*M)*(N-M+1)/M)
```

Example 54 ***SYSTAT data editor screen with commands to create confidence limits for binomial mortality data***

For statistical packages such as SYSTAT it is possible to calculate confidence limits directly using intrinsic functions for the F-distribution. Example 54 shows some tree mortality data, together with the transformation commands that would be used in the SYSTAT EDIT module to create the upper and lower confidence limit estimates. The SYSTAT FIF function returns the F-value for a given confidence level, and degrees of freedom. The symbols used in this example correspond directly to the equations discussed above, but in a real case, more meaningful symbolic names would be helpful.

4.3.5 *Regression analysis of mortality*

Mortality data can be analysed by regression using procedures similar to those applied for increment, as discussed earlier in section 4.2.

Mortality is binomially distributed and takes values between 0 and 100% for a given stand. As a dependent variable in a regression it should be transformed to reflect these conditions (Collett, 1991). This is commonly performed with the logistic transformation:

$$\mathrm{logit}(p) = \log(p/(1-p)) \qquad \{63\}$$

This gives values which vary from -∞ to +∞ as p varies from 0 to 1. The logistic function is symmetric, and gives a value of zero for mortality rates of 50%. Figure 30 shows the form of the function.

Some statistical packages can fit regressions directly to individual tree observations of death or survival, coded as 0-1 data. These use a maximum liklihood procedure, rather than conventional least squares regression analysis. Collett (1991) describes the theory to this, whilst Vanclay (1991) gives an example of an application in forestry.

However, where observations are made over variable periods, it is difficult to use this approach, and it may also be cumbersome when data is plentiful.

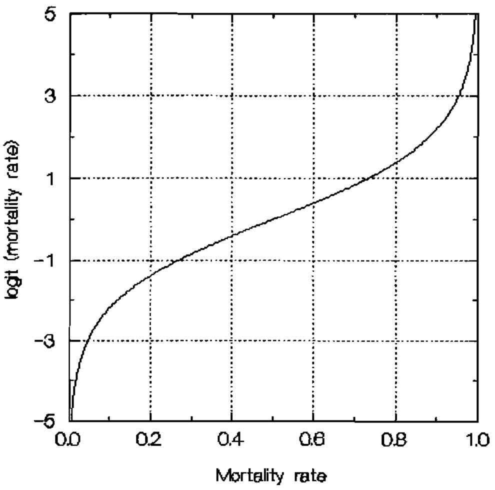


Figure 30 *Logistic tranformation function for mortality data*

The example which is discussed here demonstrates the fitting of mortality regressions to the data compiled in Example 53. Three species are extracted from the MORTSUM database for purposes of illustration, with the data listed in Example 55. The species are *Antiaris toxicaria* (ANT), *Nesogordonia papaverifera* (NES), and *Piptadeniastrum africanum* (P). The variables shown in the listing are crown position (CPOS), number of trees at the start of the period of observation (NTREES), and calculated rate of annual mortality (AMORT).

The data are shown in Figure 31, both as untransformed mortality values, and with the logistic transformation. The effect of the transformation in improving the spread of the data is apparent. Large values (15% mortality for P in crown class 1) are brought closer in to the main trend of the data, whilst the small variations in low mortalities are extended in the lower part of the graph. The functions fitted in each part of of the figure differs. That fitted to the untransformed data is:

$$\log(AMORT) = \alpha + \beta.\log(CPOS) \qquad \{64\}$$

whilst the line in Figure 31(ii) is the function:

$$\mathrm{logit}(AMORT) = \alpha + \beta.\log(CPOS) \qquad \{65\}$$

Graphically, it is not clear whether the three species may differ in their regression functions. However, this can be tested by including the species code SPP$ as a categorical variable in a general linear model. The procedure is shown for SYSTAT in Example 56. In (1), the commands used in the EDIT module are shown to create the logistic-transformed mortality, called LOGIT, and the log-transformed crown class, alled LCPOS. In (2), the MGLH module of SYSTAT is called and a linear model is fitted which includes SPP$ as a covariate. The output, shown in the inner box, gives a probability of 0.113 that the effect of SPP$ is due to chance, and hence by the usual

SPP$	CPOS	NTREES	AMORT
ANT	1	23	0.033
ANT	2	186	0.029
ANT	3	235	0.017
ANT	4	87	0.017
ANT	5	28	0.016
NES	1	553	0.053
NES	2	920	0.026
NES	3	578	0.010
NES	4	158	0.013
NES	5	21	0.009
P	1	104	0.157
P	2	409	0.031
P	3	335	0.019
P	4	62	0.013
P	5	21	0.020

Example 55 *Mortality data for 3 species from Ghana PSPs*

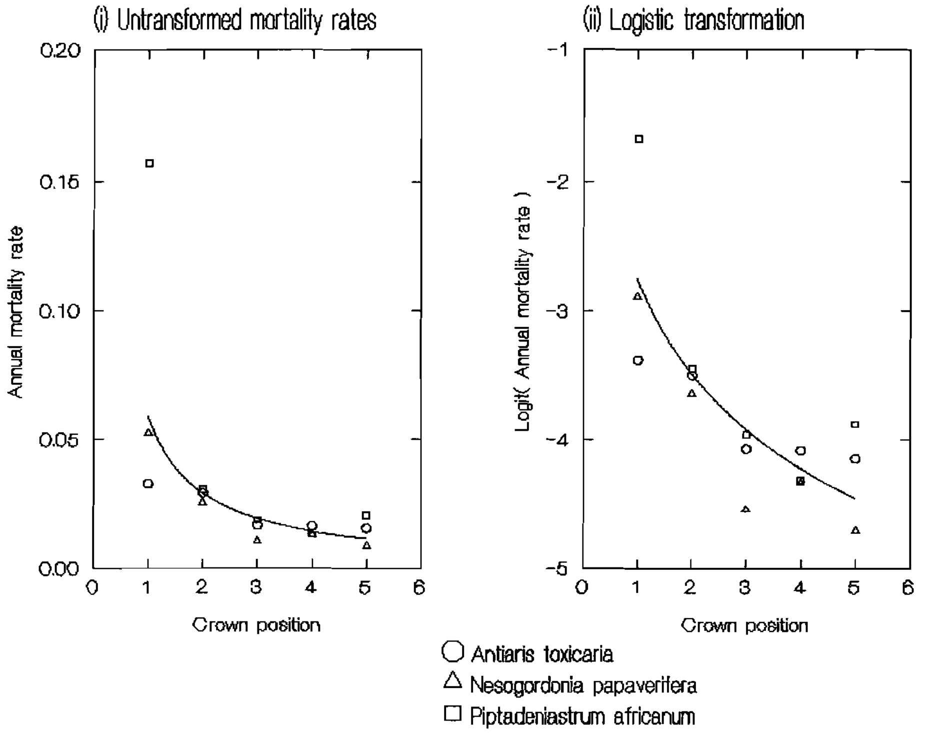


Figure 31 ***Mortality rate versus crown class for 3 species, plotted using untransformed data and the logistic transformation***

criteria, it is deemed non-significant. In (3), the same model is fitted, but without the SPP$ effect, and the coefficient values of a pooled species regression estimated.

It may be noted here that the species used in this example were selected as showing a well-defined trend from high mortality at low crown class values (shaded trees) to low mortality as exposed or emergent trees. Many species in the same dataset do not show this trend clearly, either because the data is too limited, or perhaps because they behave in contrary ways.

4.4 Regeneration and recruitment

4.4.1 *Concepts and definitions*

Regeneration is the process by which new seedlings become established in the forest. Quantitative and qualitative modelling of regeneration requires a consideration of the light requirements for germination and survival, the frequency of seeding of mother trees, seed dispersal distances, and so forth. Development of an effective regeneration model for a single species requires considerable detailed research. To do so for a large number of species, as is required for modelling the natural regeneration of tropical moist forest, is a formidable task.

Recruitment is a much more amenable process for modelling. Recruitment is defined as the number of trees which attain the minimum size for measurement over a given period. It may typically be measured in terms of numbers of trees per hectare per year (N/ha/yr). Recruitment can be observed directly on conventional PSPs, and related to stand density, logging history, forest type, presence of mother trees, and other empirically determinable factors on the plot.

Recruitment is defined relative to a specified minimum diameter for measurement. This may vary from 5-20 cm on typical PSP designs, but could conceivably be to smaller or larger limits than these. Combining data from PSPs which have been measured to different levels will probably require that the highest level is adopted for all plots in order to standardize the data. It is therefore undesirable and inefficient to economize on PSP design by having, for example, some plots measured to 20cm minimum and others to 5 cm minimum. In effect, the data on the latter group of PSPs relating to trees from 5-20 cm will be of little use and will be discarded during analysis.

There will clearly be a correlation between seedling population at a given time and subsequent recruitment levels. This correlation will be closer with a low diameter limit for recruitment, and weaker with a high diameter limit. Where regeneration data is available, this correlation can be determined empirically, and used to improve the predictive power of a growth model. The causal factors of regeneration will also therefore be linked to recruitment levels, but with a lag of some years. The time factor for this lag will depend on the mean growth rate of the seedling population and the diameter limit used to define recruitment. The higher the diameter limit, the longer the lag is likely to be. Growth rates of seedlings will generally be slower in closed, undisturbed stands, and the lag between regeneration and recruitment will correspondingly be weaker.

Vanclay (1994, chapter 10) reviews many of the approaches that have been adopted to modelling both recruitment and regeneration

4.4.2 *Causal factors and predictor variables in regression*

Causal factors which need to be considered in establishing regression models for recruitment rates will include the following:

(i) *Presence of mother trees.* For seeds with limited dormancy and dispersal, a correlation can be expected between the abundance of mother trees and recruitment. It should however be noted that logging may remove mother trees some years prior to the recruitment of established seedlings. The correlation should therefore be established between past abundance and current recruitment.

(ii) *Degree of disturbance.* Regeneration will generally follow disturbance of the stand through logging, cyclones, fires, and natural tree falls. Recruitment in completely undisturbed stands will be very slow. As the degree of disturbance increases, the species-mix of the recruitment will shift increasingly to favour pioneer species as against those of more shade-bearing disposition.

(1) Create transformed variables LOGIT and LCPOS

```
edit
  use anpmr
  let logit=log(amort/(1-amort))
  let lcpos=log(cpos)
  save anpmr
```

(2) Test if species regressions differ significantly

```
mglh
  model logit=constant+lcpos+spp$
  category spp$
  estimate
```

```
DEP VAR:   LOGIT      N:       15  MULTIPLE R: 0.886  SQUARED MULTIPLE R: 0.785


                       ANALYSIS OF VARIANCE

SOURCE        SUM-OF-SQUARES  DF  MEAN-SQUARE     F-RATIO       P

LCPOS                  5.410  1         5.410      34.747      0.000
SPP$                   0.834  2         0.417       2.678      0.113

ERROR                  1.713  11        0.156
```

(3) Fit combined regression for all species

```
model logit=constant+lcpos
est
```

```
DEP VAR:   LOGIT      N:       15  MULTIPLE R: 0.825  SQUARED MULTIPLE R: 0.680
ADJUSTED SQUARED MULTIPLE R:  .655    STANDARD ERROR OF ESTIMATE:         0.443

VARIABLE       COEFFICIENT    STD ERROR     STD COEF TOLERANCE     T   P(2 TAIL)

CONSTANT            -2.760        0.224        0.000         .   -12.328   0.000
LCPOS               -1.057        0.201       -0.825     1.000    -5.255   0.000


                       ANALYSIS OF VARIANCE

SOURCE        SUM-OF-SQUARES  DF  MEAN-SQUARE     F-RATIO       P

REGRESSION             5.410  1         5.410      27.616      0.000
RESIDUAL               2.547  13        0.196
```

Example 56 ***SYSTAT commands and output for mortality regressions, including creation of logistic transform, testing of species differences, and fitting combined regression***

(iii) *The frequency of seeds years.* Species with long intervals between seed production, coordinated mass seeding habit, and seeds of limited persistence, such as the Dipterocarpaceae will be characterised by a very variable recruitment pattern. A record of the occurrence of seed years is likely to greatly improve the correlation with recruitment. However, for the more diverse forests of West Africa or Amazonia, this is unlikely to be an important factor.

(iv) *Forest type.* A classification of forests by dominant species will greatly improve the predictability of the species mix of recruitment. In some respects, this will relate to the presence of mother trees in the population

outside the plot, and to the general occurrence of more persistent, widely dispersed seeds according to forest type. It will also relate to the suitability of climate and soils, and of conditioning factors such as fires.

4.4.3 *Modelling gross recruitment levels*

In many situations, the modelling of recruitment becomes more amenable if it is broken into two steps. In the first step, the gross number of recruits of all species is predicted from functions that may depend on basal area or stand disturbance in one or more previous time periods. In the second step, this total number can be apportioned to species groups or individual species using functions or tables.

This avoids the problem of trying to predict recruitment levels of individual species directly. This is difficult for two reasons:

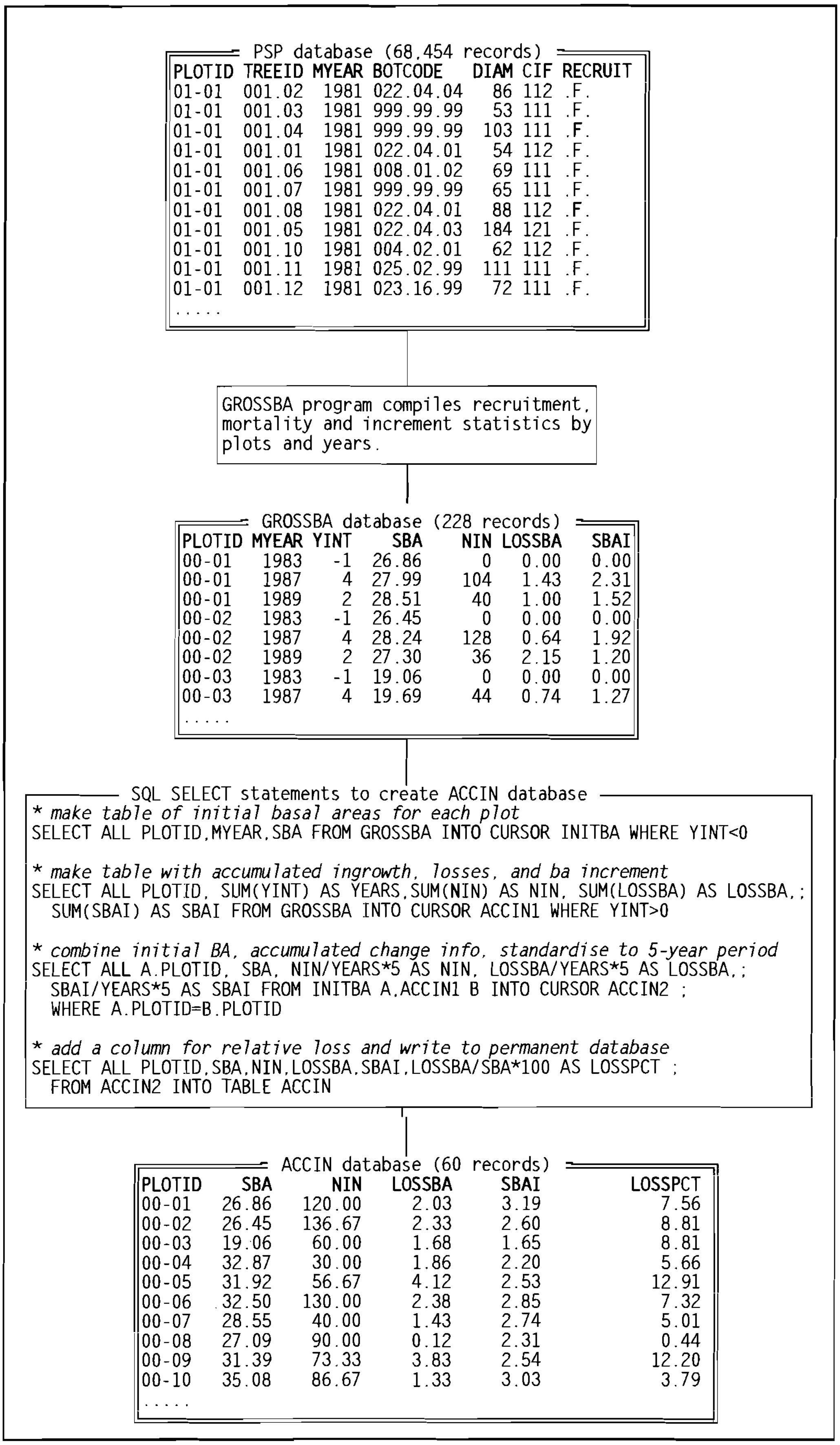
(i) *Data on an individual species will be sparse.* Regression models constructed for individual species will be poorly defined, and are quite likely to lead to pathological results.

(ii) *Achieving additivity and compatibility of species regressions is difficult.* Over the range of conditions arising in the model, the covariant behaviour of the various species occurrences needs to be reliably predicted. The total recruitment predicted by the sum of individual regressions should approximate the expected total recruitment for the stand. Achieving this kind of additivity in separate regressions is very complex and may be impossible with many species each represented by sparse data.

This 2-step approach to recruitment modelling was adopted by Vanclay (1989a) for the North Queensland rainforest, and is also exemplified in the CAFOGROM case study.

Vanclay (1994:198 *ff.*) also describes a 2-stage approach to recruitment modelling, which may be distinguished from the *2-step* method described above. The 2-stage approach is particularly appropriate for stochastic models, and uses an initial function which determines whether or not recruitment will occur in any one year for a particular species. If recruitment occurs, the second stage of the model can then be called to estimate quantities of ingrowth. The 2-stage approach is likely to be well-suited to trees with long intervals between flowering, such as many Dipterocarps. It is also more appropriate to modelling regeneration (seedling establishment) rather than recruitment (ingrowth of pole-sized trees).

Gross recruitment rates will be generally associated with past disturbance as a result of logging, natural mortality, storms, fires, or other factors. It is logical to expect a consistent relationship between the basal area removed and the number of trees recruited after a lag of 5-10 years. This is because basal area losses correlate directly with crown area lost, and hence growing space created. Recruits, by definition, are of similar size (close to the specified threshold diameter), and hence on average will have a similar growing space occupance.



```
═══ PSP database (68,454 records) ═══
PLOTID TREEID MYEAR BOTCODE    DIAM CIF RECRUIT
01-01  001.02  1981 022.04.04   86 112 .F.
01-01  001.03  1981 999.99.99   53 111 .F.
01-01  001.04  1981 999.99.99  103 111 .F.
01-01  001.01  1981 022.04.01   54 112 .F.
01-01  001.06  1981 008.01.02   69 111 .F.
01-01  001.07  1981 999.99.99   65 111 .F.
01-01  001.08  1981 022.04.01   88 112 .F.
01-01  001.05  1981 022.04.03  184 121 .F.
01-01  001.10  1981 004.02.01   62 112 .F.
01-01  001.11  1981 025.02.99  111 111 .F.
01-01  001.12  1981 023.16.99   72 111 .F.
.....
```

```
GROSSBA program compiles recruitment,
mortality and increment statistics by
plots and years.
```

```
═══ GROSSBA database (228 records) ═══
PLOTID MYEAR YINT   SBA    NIN LOSSBA  SBAI
00-01   1983   -1  26.86     0   0.00  0.00
00-01   1987    4  27.99   104   1.43  2.31
00-01   1989    2  28.51    40   1.00  1.52
00-02   1983   -1  26.45     0   0.00  0.00
00-02   1987    4  28.24   128   0.64  1.92
00-02   1989    2  27.30    36   2.15  1.20
00-03   1983   -1  19.06     0   0.00  0.00
00-03   1987    4  19.69    44   0.74  1.27
.....
```

```
───── SQL SELECT statements to create ACCIN database ─────
* make table of initial basal areas for each plot
SELECT ALL PLOTID,MYEAR,SBA FROM GROSSBA INTO CURSOR INITBA WHERE YINT<0

* make table with accumulated ingrowth, losses, and ba increment
SELECT ALL PLOTID, SUM(YINT) AS YEARS,SUM(NIN) AS NIN, SUM(LOSSBA) AS LOSSBA,;
  SUM(SBAI) AS SBAI FROM GROSSBA INTO CURSOR ACCIN1 WHERE YINT>0

* combine initial BA, accumulated change info, standardise to 5-year period
SELECT ALL A.PLOTID, SBA, NIN/YEARS*5 AS NIN, LOSSBA/YEARS*5 AS LOSSBA,;
  SBAI/YEARS*5 AS SBAI FROM INITBA A,ACCIN1 B INTO CURSOR ACCIN2 ;
  WHERE A.PLOTID=B.PLOTID

* add a column for relative loss and write to permanent database
SELECT ALL PLOTID,SBA,NIN,LOSSBA,SBAI,LOSSBA/SBA*100 AS LOSSPCT ;
  FROM ACCIN2 INTO TABLE ACCIN
```

```
═══ ACCIN database (60 records) ═══
PLOTID   SBA     NIN   LOSSBA    SBAI        LOSSPCT
00-01   26.86  120.00    2.03    3.19           7.56
00-02   26.45  136.67    2.33    2.60           8.81
00-03   19.06   60.00    1.68    1.65           8.81
00-04   32.87   30.00    1.86    2.20           5.66
00-05   31.92   56.67    4.12    2.53          12.91
00-06   32.50  130.00    2.38    2.85           7.32
00-07   28.55   40.00    1.43    2.74           5.01
00-08   27.09   90.00    0.12    2.31           0.44
00-09   31.39   73.33    3.83    2.54          12.20
00-10   35.08   86.67    1.33    3.03           3.79
.....
```

Example 57 ***Preparing a database of gross recruitment for analysis from raw PSP data***

Therefore the number of recruits will be proportional to the growing space available. Given information on crown dimension/dbh relations of common species, it will be possible to determine approximately the total number of recruits.

To illustrate typical analysis procedures and regression results, some data is used from a logging experiment at Km 114, Tapajos, Brazil. Some background to this experiment is described in Silva et al. (1994). The PSP measurement procedures and information coding standards are given in Korsgaard (1993). The data presented here is incomplete and is intended only for purposes of illustrating the analytical procedures.

The data manipulations start from the file of raw PSP data, shown in Example 57. This comprises one record per tree. Each record contains the following fields:

PLOTID — Plot identification number. This is a two part code, with the first two digits corrsponding to the treatment, and the second two to the replicate.

TREEID — The first three digits refer to the quadrat number (10 x 10 m), and the second two to the tree number.

MYEAR — Year of measurement.

BOTCODE — A hierarchical botanical code using family, genus and species code. Unknown genera or species are indicated by 99.

DIAM — Tree diameter in mm. The lower limit for measurement was 5 cm.

CIF — A stem condition code, as detailed in Korsgaard (1993). Codes other than 111 or 112 are effectively dead, fallen, broken or missing.

RECRUIT — A field added by the analyst, used to flag recruit trees. The method for filling this field is detailed in Example 60.

The PSP data is compiled by a program called GROSSBA into a database of the same name. GROSSBA assumes two prerequisite operations:

(i) The database must be indexed, using the command:

```
INDEX ON PLOTID+TREEID+STR(MYEAR,4) TAG TREE
```

This index brings together records for the same tree in their correct sequence.

(ii) A subsidiary database YPSP of first and last measurement years for each plot is created with the SQL command:

```
* Program GROSSBA:  compiles initial basal area, basal area growth, losses, and recruitment by plots
parameters psp
set talk off
set safety off
public plotid0, treeid0, diam0, cif0, year0, ylast0
* create arrays to store recruits, losses, gross ba in each year by plots
select distinct plotid from (psp) into array plots
select distinct myear from (psp) into array years
mp=alen(plots)
my=alen(years)
declare recruits(mp,my),losses(mp,my), grossba(mp,my), bainc(mp,my)
for np=1 to mp
  for ny=1 to my
    store 0 to recruits(np,ny),losses(np,ny),grossba(np,ny),bainc(np,ny)
  next
next
* open data files with appropriate index tags and set relational linkage
close databases
use ypsp in 0 order tag plotid
use (psp) in 0 alias psp order tag tree
select psp
set relation to PLOTID into ypsp
*  loop through data grouped by plots
do while .not. eof()
  * Reset plot and tree ID at start of new plot.
  plotid0=PLOTID
  np=ascan(plots,PLOTID)
  *  loop through data grouped by trees
  do while PLOTID=plotid0 .and. .not. eof()
    ny=ascan(years,MYEAR)
    do first
    *  check if tree is a recruit when first measured
    if MYEAR>ypsp.yfirst
      recruits(np,ny)=recruits(np,ny)+1
    endif
    *  loop through remeasurements for a tree
    do while TREEID=treeid0 .and. .not. eof()
      ny=ascan(years,MYEAR)
      if .not. inlist(CIF,[111],[112]) .and. inlist(cif0,[111],[112])
        losses(np,ny)=losses(np,ny)+DIAM^2
      else
        grossba(np,ny)=grossba(np,ny)+DIAM^2
        bainc(np,ny)=bainc(np,ny)+(DIAM^2 - diam0^2)
      endif
      do first
      * move to next record in file
      skip
    enddo
  enddo
enddo
* write results from array to file, adjusting from raw Σd² in mm to m²/ha.
psize=0.25
mm=1000
mha=pi()/(4*mm^2*psize)
create table grossba (PLOTID C(5),MYEAR N(4),YINT N(2),SBA N(6,2),NIN N(6,0),LOSSBA N(6,2),SBAI N(6,2))
for np=1 to mp
  for ny=1 to my
    if grossba(np,ny)>0 .or. losses(np,ny)>0
      yint0=iif(ny>1, years(ny)-years(ny-1),-1)
      append blank
      replace PLOTID with plots(np), MYEAR with years(ny), YINT with yint0, SBA with grossba(np,ny)*mha,;
              NIN with recruits(np,ny)/psize, LOSSBA with losses(np,ny)*mha, SBAI with bainc(np,ny)*mha
    endif
  next
next
return
```

Example 58 *Program to compile recruitment numbers by plots and measurement years*

```
SELECT ALL PLOTID, MIN(MYEAR) AS YFIRST,MAX(MYEAR) AS YLAST;
        FROM (PSP) INTO TABLE YPSP GROUP BY PLOTID
```

This database should be indexed on PLOTID after it has been created by the SQL statement.

The output database created by GROSSBA is defined in line 55 of the program, with a partial listing shown in Example 57. For each plot and measurement year, it contains the following fields:

YINT Interval in years from the preceding measurement. This is given as -1 for the first measurement in a sequence.

SBA Standing basal area (m²/ha) at the start of a period.

NIN Number of recruit trees (N/ha) over a period.

LOSSBA Basal area of trees lost through mortality, crown breakage or harvesting (m²/ha).

SBAI Basal area increment of live trees, excluding new recruits, over the period (m²/ha).

The program is complicated by the need to store intermediate results in arrays. The following is a discussion of the principle features:

Program lines *Description*

1-5 The name of the PSP file is passed as a parameter to the program. Interactive responses are suppressed (SET TALK), including checks on files and index tags overwritten or erased (SET SAFETY). Variables first referenced in the FIRST subprogram are declared PUBLIC so that they can be used outside the subroutine.

6-16 This section creates two arrays: PLOTS and YEARS, which respectively list the plot IDs and years occurring in the data set. These arrays are generated by the SQL queries at lines 7 and 8. Arrays to store recruits, losses, gross basal area, and basal area increment are then created and initialized to zero.

17-22 Any existing databases are closed. The YPSP and PSP databases are opened, and a relationship based on PLOTID established between them. This allows the program, as it scans records in the PSP file, to check from YPSP whther the measurement year may be the first or last year.

24-39 Three nested loops are involved in scanning and accumulating data. The outer one considers data grouped by plots. It repeats each time there is a change in PLOTID, and terminates at the end of the file. The index number for the current plot is determined at line 30 from the list of plot

IDs in array PLOTS. The second level loop scans through trees within a plot, and repeats each time there is a change in the TREEID. It terminates when the PLOTID changes.

The first measurement for a tree is checked at line 33 to see if it is a recruit. This is defined when the measurement year of first occurrence is greater than the first measurement year for the plot, determined by reference to the YPSP file. If the tree is a recruit, it is counted in the appropriate array cell, indexed by plot and year.

The procedure FIRST, shown in the box opposite, initializes a pair of tree records. It is called for the initial measurement at line 31, and subsequently for each successive pair of records at line 45.

```
procedure first
* Saves details of the first
* of a pair of tree records
treeid0=TREEID
diam0=DIAM
cif0=CIF
year0=MYEAR
return
```

Example 59 *GROSSBA procedure FIRST*

39-44 Accumulates results for losses (dead or harvested trees) or for living trees. The CIF code is compared for the current and previous record in a tree. If they are not both 111 or 112, then the tree is dead, damaged or missing, and its raw D^2 is accumulated in the appropriate cell of the LOSSES array.

Otherwise, the tree remains alive. Standing basal and basal area increment are totalled by accumulating raw D^2 and the change in D^2 over the period.

52-54 Constants used to convert raw accumulated D^2 per plot to m^2/ha are set here. PSIZE is plot size in ha. MM is the number of diameter measurement units (mm) per metre. MHA is the calculated conversion constant.

55-65 The output file is created, and the array cells written to it. Only non-zero cells are written out (line 58). The data is converted from raw accumulated values to standard units as it is output (lines 61-62).

The GROSSBA program summarizes the essential recruitment information, but the output requires some further manipulation to prepare it for regression analysis. This has been done in this example through a series of SQL queries, shown in Example 57. The purpose of these is to generate the file ACCIN, in the format shown. For each plot, this gives the standing basal area, number of recruits, basal area loss, and basal area increment. Basal area loss is also expressed in relative terms as LOSSPCT, defined as (LOSSBA/SBA) x 100. The observations are standardised to a 5-year period from the variable periods for the separate plot measurements.

The ACCIN file was imported into SYSTAT, and the relationship between recruitment (NIN) and other variables examined graphically and by regression. The best fit was obtained between NIN and LOSSPCT with a simple linear regression of the form:

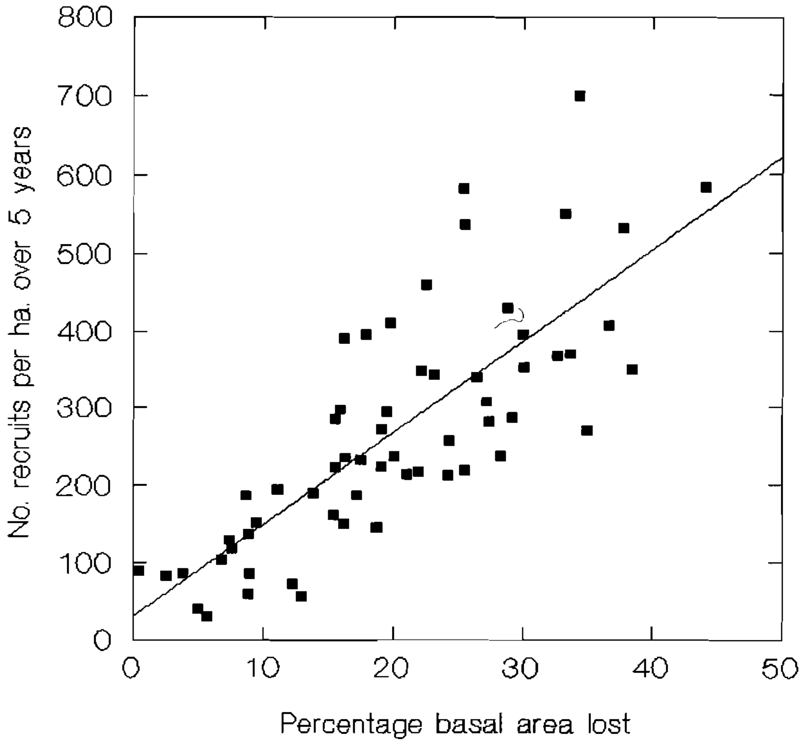


Figure 32 ***Gross recruitment as a function of percentage basal area lost over a period***

$$\mathrm{NIN} = 31.8 + 11.8\ \mathrm{LOSSPCT} \qquad (R^2\ 0.622) \qquad \{66\}$$

In this equation, the constant (31.8) was not significantly different from zero, and as a rule of thumb for the region in question (*Terra firme* rainforest, lower Amazon), it would appear that the number of ingrowth trees to 5 cm dbh over a 5 year period is approximately 10 times the percentage basal area lost through harvesting or mortality over the 5-year period. The data and the fitted function are shown in Figure 32.

Equation {4.4.1} can be used to predict total number of recruits over a period. It remains necessary to determine the species composition of those recruits with different intensities of disturbance. The procedures described above will obviously vary depending on the field names of the database and methods for coding mortality, damage, harvesting, and so on. There are also several pathways by which the same end result may be achieved. The form of the gross recruitment function will also depend on forest type, and on features of the dataset. In this case, logging experiment data from a relatively compact area was used, giving a wide coverage of stand treatments, but with limited forest type variation. If regular PSPs were used, the converse would probably be true, with more forest type variation, but greater similarity of treatments. In Dipterocarp forests, the periodic nature of the recruitment process would also tend to obscure or weaken correlations with stand treatment over short periods.

4.4.4 *Species mixture of recruitment*

A model that is to be effective for tropical forest management should consider the changes in speciation that are likely to occur as utilization becomes progressively heavier. In undisturbed forest, only shade-tolerant species are likely to regenerate. In logged forest, pioneer and light-demanding species will be encouraged. This process will be accentuated with heavier logging.

A study by Hawthorne (1993) of forest regeneration after logging in Ghana, exemplifies some of the techniques and methods that may be adopted to determine the regeneration preferences of species. However, given complete PSP data of the type described in Example 57, it is possible to build lists of species recruitment probabilities directly from the PSP data without prior knowledge or independent studies of this kind.

The objective is to develop a series of lists, each relating to a different level of disturbance. Each list gives the proportion of total recruitment to be allocated to a particular species.

```
* Program FLAGIN:  sets RECRUIT flag True for ingrowth trees in PSP database
parameters psp
set talk off
set safety off
* open data files with appropriate index tags and set relational linkage
close databases
use ypsp in 0 order tag plotid
use (psp) in 0 alias psp order tag tree
select psp
set relation to PLOTID into ypsp
*  loop through data grouped by plots
do while .not. eof()
  * Reset plot and tree ID at start of new plot.
  plotid0=PLOTID
  *  loop through data grouped by trees
  do while PLOTID=plotid0 .and. .not. eof()
    treeid0=TREEID
    *  check if tree is a recruit when first measured
    if MYEAR>ypsp.yfirst
      replace RECRUIT with .T.
    endif
    *  loop through remeasurements for a tree
    do while TREEID=treeid0 .and. .not. eof()
      * move to next record in file
      skip
    enddo
  enddo
enddo
set talk on
return
```

Example 60 *Program to mark recruit trees in a PSP file*

The first step is to flag recruit trees in the original PSP database. For this purpose, a field is provided called RECRUIT (see page 150) which is originally empty. A program FLAGIN, shown in Example 60, scans the database and fills RECRUIT as TRUE or FALSE depending on whether or not the tree may be regarded as a recruit. It will be noted that the logic of this program is a subset of that in the GROSSBA program discussed on page 152. The same structure of three nested loops is employed to scan through plots, trees, and measurement pairs. As with GROSSBA, a relational linkage is established to the pre-defined YPSP database. The first measurement of each tree is checked for its date. If it occurs later than the first measurement of the plot, then the tree is flagged as a recruit. The inner loop through subsequent measurements for the tree is empty except for the SKIP function to move the database record pointer.

The remaining stages in defining recruitment probabilities can then be performed by a series of SQL queries, as follows:

-
  ```
  SELECT ALL PLOTID, IIF(LOSSPCT<20,1,2) AS LOSSK ;
          FROM ACCIN INTO CURSOR LPC
  ```

  This creates a list of plots with their disturbance class in a temporary database called LPC (Loss percent Class). The IIF() function classifies plots such that those with less than 20% basal area loss are class 1, and others are class 2. The 20% threshold was selected as splitting the plots into approximately equal groups.

  Replacing the IIF() function by one such as INT(LOSSPCT/10)+1 would divide the plots into 10% loss classes, coded from 1 to 10 in the field LOSSK. The disadvantage is that for a limited number of plots (in this case 60 of 0.25 ha), such a refined breakdown gives two few observations in each category to give a consistent gradation of recruitment probabilities. By nesting IIF() functions, multiple classes based on arbitrary intervals can be defined. For example:

  ```
  IIF(LOSSPCT<15,1, ;
          IIF(LOSSPCT<40,2,3))
  ```

  would give class 1 as loss% <15, class 2 as 15%-39%, and class 3 as ≥40%.

```
BOTCODE    LOSSK  PROBIN
004.02.01    1    0.0370
004.02.01    2    0.0183
005.01.01    1    0.0497
005.01.01    2    0.0928
006.01.01    1    0.0274
006.01.01    2    0.0337
009.01.06    1    0.0511
009.01.06    2    0.0416
016.01.99    1    0.0278
016.01.99    2    0.0221
021.99.99    1    0.0228
021.99.99    2    0.0180
023.16.99    1    0.1455
023.16.99    2    0.1407
029.04.01    1    0.0328
029.04.01    2    0.0678
029.04.02    1    0.0347
029.04.02    2    0.0974
041.99.99    1    0.0374
041.99.99    2    0.0291
047.01.01    1    0.0607
047.01.01    2    0.0382
```

Example 61 ***Listing of selected species from PROBIN database with high recruitment probabilities***

-
  ```
  SELECT ALL A.BOTCODE, B.LOSSK, CNT(RECRUIT) AS NIN ;
          FROM (PSP) A,LPC B INTO CURSOR RAWIN ;
          WHERE A.PLOTID=B.PLOTID AND A.RECRUIT ;
          GROUP BY A.BOTCODE,B.LOSSK
  ```

  This creates a temporary table called RAWIN of raw recruitment numbers over the total interval of measurement. The output file contains the botanical code of the species (BOTCODE), the loss % class (LOSSK), and the raw recruitment count (RAWIN).

-
  ```
  SELECT ALL LOSSK, SUM(NIN) AS TOTIN FROM RAWIN INTO CURSOR TOTIN ;
          GROUP BY LOSSK
  ```

  This creates a table showing the total recruitment numbers in each loss% class, for all species pooled. This tempory table is called TOTIN.

-
  ```
  SELECT ALL A.BOTCODE, A.LOSSK, A.NIN/B.TOTIN AS PROBIN ;
          FROM RAWIN A, TOTIN B INTO TABLE PROBIN WHERE A.LOSSK=B.LOSSK
  ```

  This creates the final table PROBIN of recruitment probabilities from the raw recruitment numbers in RAWIN, divided by the totals in TOTIN. There are three output fields: Botanical code (BOTCODE), Loss% class (LOSSK), and probability of recruitment (PROBIN).

A selection of values for some species with high recruitment probabilities are shown in Example 61 to illustrate the structure of PROBIN and the differential effects of the different disturbance levels.

This data is presented graphically in Figure 33. Species names have been substituted for the botanical codes, using further database query commands, and the resultant file exported to a graphics package. The graph shows that many of the commonly recruited species are favoured by higher levels of disturbance. *Cecropia* species and *Jacaranda copaia* are typical pioneers. On the other hand, *Duguetia echinophora* and *Rinorea flavescens* regenerate more heavily in the less disturbed forest.

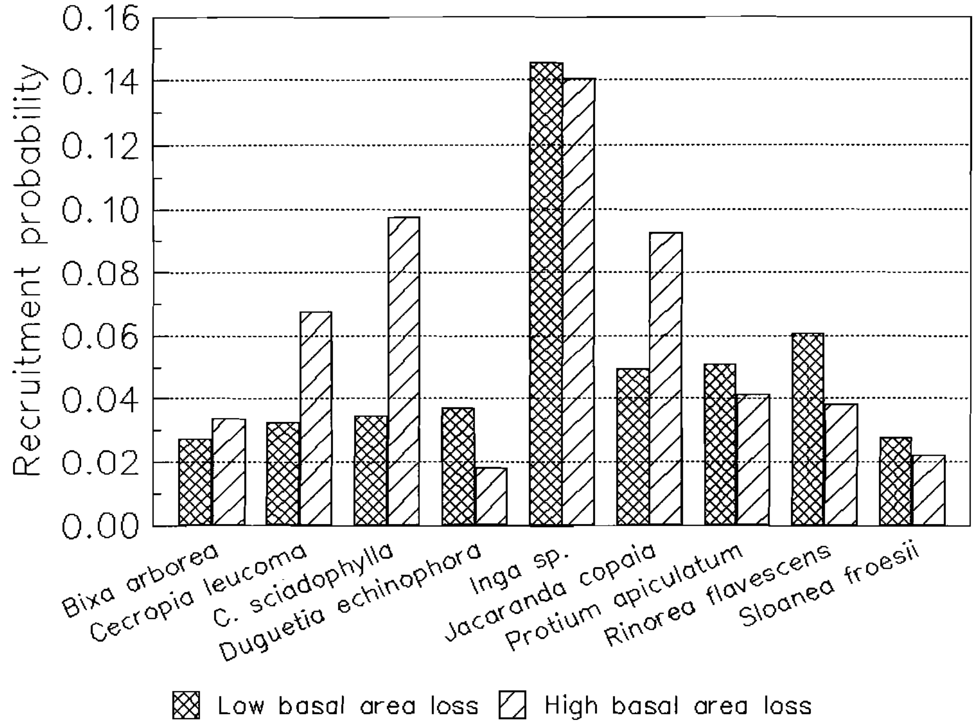


Figure 33 *Recruitment proportions of common species on the logging experiment at Km 114, Tapajos, Brazil*

4.5 Harvesting, silvicultural treatment, and logging damage

4.5.1 *Specification of harvesting and treatment rules*

The specification of rules for harvesting and associated silviculture in mixed tropical forests is complex, and many variations are possible. The most basic elements are generally felling cycle and a species-determined minimum felling diameter. The following is a list of criteria that may be applied in varying combinations:

Felling cycle: This is the period that elapses between successive harvesting operations. Whilst harvesting is commonly specified on the basis of felling cycle, there is no *a priori* necessity for this. It would be equally reasonable to specify felling when stands reach a certain basal area, for example, with a variable felling cycle between stands.

Minimum felling diameter (MFD): This is generally a species-group dependent parameter, and controls the smallest tree that may be felled in that group.

Number of retained trees: A given number of trees may be retained above the MFD for a given species. This can be a silvicultural measure, to promote seed production, and a conservation measure, to ensure valuable trees or those of ecological importance are not logged to extinction.

Percentage of retained trees: This has a similar but weaker effect to retaining an absolute number of trees per unit area above the MFD.

Percentage felled trees by size class: This is a complex rule to apply in mixed tropical forest. It is practiced in selection forest, where a specified proportion of removals from each class can be calculated to give a stable age-class structure using a matrix model (Usher, 1966, 1969).

Adequacy of advance growth: In Malaysian and Indonesian management systems, the selection of the felling rules is determined on the basis of the presence of an adequate stock of pole sized advance growth (Poore et al., 1989).

Understorey condition: In Ghana, a system has recently been adopted of scoring compartments according to indicators of fire damage and excessive canopy opening. Logging is only permitted in stands which are in good condition (Maginnis, 1994).

Rules which are based on the mensurational parameters of the tree growing stock can be readily incorporated into a growth model. Those that rely on subjective indicators, such as the Ghana understorey condition scoring, are more difficult to accomodate as endogenous factors.

The most common rules, and perhaps the most practical to implement and enforce, involve the combination of felling cycle, minimum felling diameter, and numbers of retained trees above the minimum diameter. The programming example will consider how these can be used in a cohort model.

Silvicultural treatments that involve the poisoning or girdling of trees that are competitors to commercial species can be accomodated directly in tree-position models. One of the strengths of this class of model is the facility with which crown thinning may be modelled. In a cohort or other non-spatial model, empirical information about the size distribution of trees marked for thinning is required. This can then be simulated as mortality on the specified size and species distribution. These empirical distributions can be determined from static sampling, and do not require PSP measurements.

Silvicultural treatments that influence non-tree components, such as fire control and climber cutting cannot be modelled directly in a standard model. Indirectly, categorical variables may be used in some growth functions to indicate the levels of treatment applied, provided empirical data is available to distinguish the effects of treatment.

4.5.2 *Normative and empirical simulation of harvesting*

Normative simulation of harvesting constitutes the application of given felling rules in a full and complete manner, without consideration of the variable effects of markets, logistics, and other factors that may lead to either incomplete or excessive felling of some categories of trees.

Empirical simulation of harvesting requires statistical information on species and sizes actually removed from the forest. It can reflect the fact that some 'commercial species' are in fact in little real demand. It can also reflect the unauthorised logging of trees in protected categories or below the minimum felling diameter.

A normative harvesting model therefore specifies what should happen, whereas an empirical one indicates what actually does happen.

A normative model is readily developed, using the logical application of a given set of felling rules. The normative approach is appropriate for the preparation of felling plans and estimates of allowable cut and sustained yield.

An empirical model requires statistical data on trees actually removed from the forest, in terms of species and size distribution. This may be obtained from conventional trade statistics in some countries. In other cases, it may require specific monitoring studies or pre- and post-harvesting inventories. An empirical approach is appropriate when attempting to project actual output from a forest area, or when attempting to determine the long-term sustainability or effects of current pattern of operations.

The distinction between the normative and empirical situation can be very considerable. There are many species which are partially commercial, which may be listed as part of the yield or planned output from the forest, but which cannot in fact be absorbed by markets. A comprehensive study should examine both normative and empirical outputs, and use the comparative information to improve controls on forest management. Such a comparative study can also focus attention on species which should be promoted, and others which should be protected.

Vanclay (1989b) demonstrated an approach to the development of empirical functions for the probability of harvesting a stem, based on species, size, and time since the last felling operation. Alder (1990) used trade statistics to determine the percentage of mature trees likely to be removed in particular species groups. In the CAFOGROM model discussed in the case study, very limited data was available on empirical effects of harvesting, and a primarily normative approach was adopted.

4.5.3 *Modelling logging damage*

Logging damage is an important consequence of harvesting, and one which influences future stand growth and yield. PSP data can be used to show how mortality changes following logging, or on plots with different logging intensities. The effects are likely to be clearest if the PSPs form part of a logging experiment with a wide range of carefully specified treatments. Monitoring PSPs may not give a clear record of the timing or nature of logging.

Logging damage to individual trees may be of varying severity, and will translate into short- or long-term mortality to a degree that depends on the characteristics of the species, especially the decay-resistance of the timber. However, Tang (1976) found that mortality on undamaged trees, especially in the smaller sizes, also rises markedly following logging. Oliver & Larson (1990:87) discuss the physiological basis for mortality following release from shade. The consequences of these phenomena are that short-term studies of logging damage are much less useful than long-term PSP observations of mortality. The establishment of a regression model linking damage of varying severity and species to long-term survival can allow the wider use of short-term studies in comparing, for example, logging with different technical systems (tracked and rubber-tyred skidders, cable logging, *etc.*).

(i) Years after logging

Year	Mortality%
1	4.16
2	9.53
3	4.02
4	3.90
5	2.41

(ii) Type of damage

Type	% of mortality
None	33
Stem	6
Crown	23
Stem+Crown	38

Table 10 *Distribution of mortality, Trengganu cutting trial*

Figure 34 shows the results obtained by Tang (1976). Mortality in the 5 years following logging is a linear function of the intensity of logging, measured as the percentage of original basal area removed. Tang found that mortality peaked in the second year after logging, with annual rate of 9%, an declined regularly thereafter (Table 10 *(i)*). Highest mortality was on trees with both crown and stem damage (38%), but 33% of mortality was on completely undamaged stems (Table 10 *(ii)*).

In a growth model, the method of applying these kinds of results depends on the time period of the simulation. It appears that if a 5-year time step is used, post-logging mortality can be assigned directly in the same period that logging is performed. If a shorter interval is used, or if post-logging mortality appears to persist at high levels for longer periods, then an exponential decay model will be required to reduce the residual mortality by a constant fraction for each period following logging.

4.5.4 *Forest disturbance patterns*

Following logging, the forest can be partitioned into a number of recognisable zones, including undisturbed forest, skid trails, felled tree gaps, log landings, and haulage roads. The area attributable to each of these situations will depend on both the intensity and technical method of the logging. Examples of some studies are Nicholson (1958), Jonkers (1987), Schmitt & Bariteau (1989), Hawthorne (1993).

Jonkers (1987) used regression analysis to show the proportions of the stand falling into the categories of tree fall gaps and skid trails as a function of the basal area removed from plots on a logging experiment. His results are shown in Figure 35.

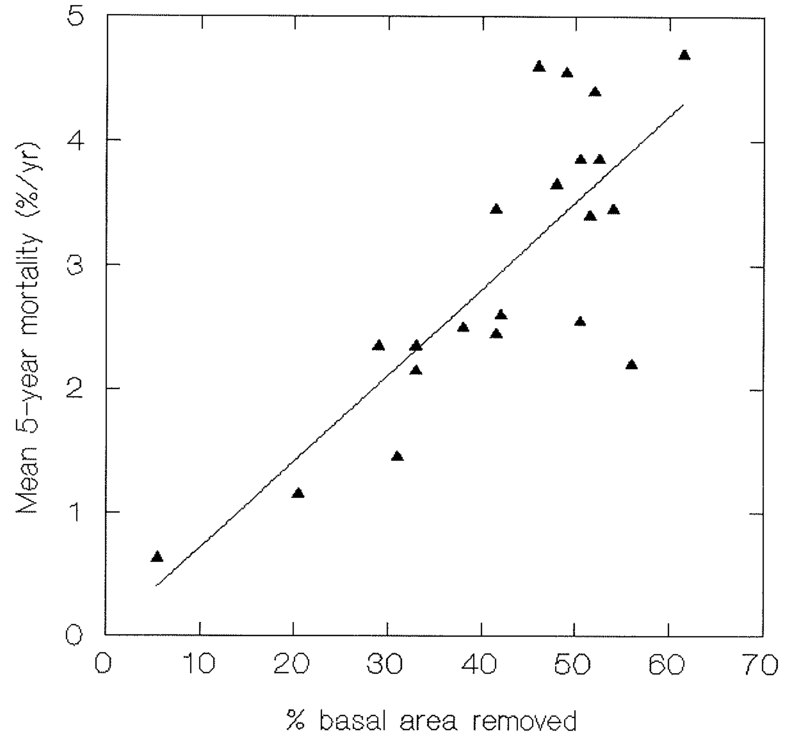


Figure 34 *Post-logging mortality as a function of percentage basal area removed, Trengganu cutting trial, W. Malaysia (from Tang, 1976, fig. A-10)*

The lines represent power functions of the form:

$$Y = \alpha . X^{\beta}$$

The best correlations are with undisturbed forest and skid trails ($R^2 \approx 0.7$); that with gaps is relatively weak ($R^2 \approx 0.22$). Since the three lines must always sum to 100%, only two of them need be fitted; the coefficients and form of the third are then determined algebraically.

If recruitment data are measured on PSP quadrats which are characterised at each remeasurement as forming one of the disturbance categories, then it is possible to analyse recruitment making use of the different ecological features of these niches. Hawthorne's (1993) study of regeneration after logging in Ghana, for example, provided typical lists of species that could be used to build a recruitment probability list of the type discussed in section 4.4.4, page 155 *ff*.

In order to make effective use of such information, the simulated forest units (SFU's, see page 101 *ff*.) need to be partitioned by the disturbance categories. This would be feasible in a sophisticated cohort model. The cohort data structure would require an additional field, denoting the disturbance category, say DISTK. For each SFU, the relative proportion that would fall into each category would be calculated from regressions of the type illustrated in Figure 35, depending on the basal area removed or logging intensity. The cohort list for the SFU would then be copied for each disturbance category. It would be modified as would be expected by the logging process. Thus on skid trails, most of the trees would be

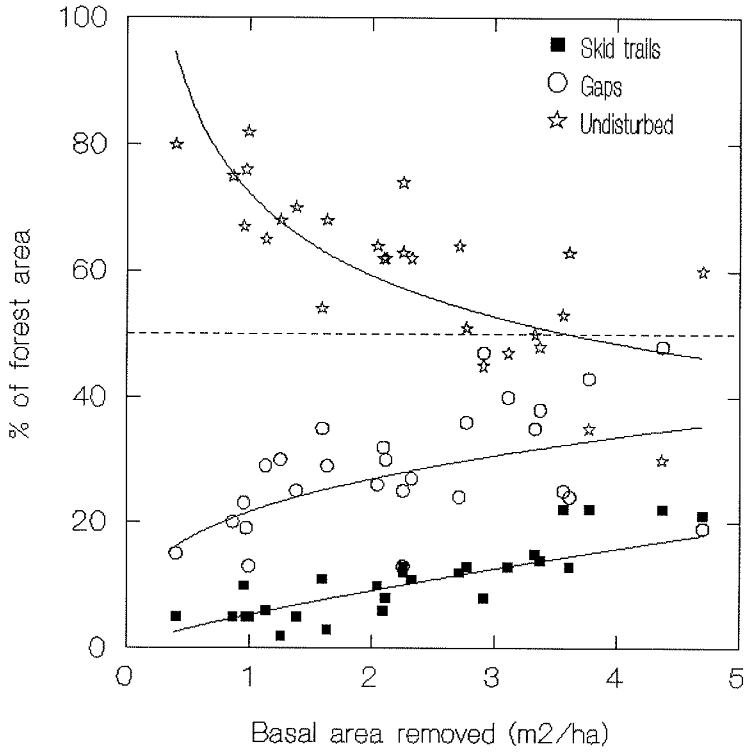


Figure 35 *Percentage of forest in skid trails and gaps, MAIN experiment, Surinam (after Jonkers, 1987)*

obliterated. In gaps, there would be high but partial mortality and removal, including especially the subject trees. Undisturbed forest would not be modified. The sub-SFUs would then enter the recruitment process using differential species lists. Each sub-SFU would be tracked independently thereafter, and a list would be maintained of sub-SFU areas.

Whether this approach would yield better predictions than a simpler one in which a stand as a whole is characterised by a general disturbance level, directly indicated by logging intensity, is not known. It remains a topic that will be interesting for future research.

Spatial modelling strategies such as gap and tree position models do need to consider the detailed features of disturbance caused by logging. In a gap model, the stand units can be assigned to gaps or undisturbed forest, etc. according to whether they contain a subject tree to be logged. Location on skid trails may be determined by two routes, depending on model design:

- If the gap units have an explicit spatial relationship, skid trail locations may be simulated;
- If gaps are spatially independent, then assignment to a skid trail may be based on a Monte Carlo function using the proportion of forest affected.

4.5.5 *Harvesting procedure design for a cohort model*

The basic logic of a procedure to simulate harvesting can be illustrated by the *Harvest* procedure in the CAFOGROM listing (lines 493-563, page 205 *ff.*). This is written in C and will be obscure to those readers not familiar with the language. The logical steps involved are as follows:

(i) *Select and index eligible cohorts:* The program will scan through the SFU array for the stand to be harvested, marking those cohorts whose species or size characteristics renders them eligible for harvesting. Typically, they must be above a minimum permitted size, and of a commercial species group.

In the simplest type of model, where all eligible trees are felled, this is all that is required. In CAFOGROM, a concept of volume control is used, whereby removals will be restricted to a percentage of total volume, or to an absolute volume/ha. For this, the list of eligible trees are sorted, from largest to smallest.

These steps are accomplished in the CAFOGROM listing at lines 510-528.

(ii) *Remove eligible trees and accrue statistics:* The eligible trees are removed by scanning through the cohorts, using the index of selected trees to pick the appropriate entries. The stocking of the harvested cohort is set to zero, and at the same time, statistics for removed volumes and basal areas are accumulated.

This is performed in CAFOGROM at lines 533-547. *hVol* is the accumualted harvested volume (summed at line 540).

(iii) *Adjust stocking of residual trees for mortality.* An empirical function is used to determine mortality levels on remaining trees, and cohorts are scanned and their stockings adjusted to account for damage. This is done in CAFOGROM at lines 548-559.

A more sophisticated concept, and one the author is likely to implement in future versions of cohort models, would be to partition the entire SFU array into sub-stands representing three phases of logging damage: Undisturbed forest, felling gaps and skid trail margins where large trees may be damaged by seedlings and soil is largely intact, and the totally destroyed surface of skid trails, roads and landings. Each of these would then follow a separate regeneration and recruitment pathway.

4.5.6 *Simulation of liberation and refining treatments*

In concept, the simulation of liberation and refining treatments is very similar to that of harvesting. A list of eligible cohorts is selected according to appropriate criteria. It is sorted if controls such as basal area limits are to be applied. The stocking of selected cohorts is then reduced to zero to reflect the girdling, poisoning or cutting of such trees.

The main variations will be in the method of tree selection. For a refining operation, or systematic thinning, all trees which meet specified size and species criteria will be selected. This is the method adopted in CAFOGROM (procedure *Thinning*, lines 565-604, page 205 *ff.*).

Liberation thinning cannot be precisely simulated in a cohort model, as there is no information on the relative position of trees. It could be approximated by selecting trees of non-commercial species with high crown status, but this will not allow two common situations to be considered. One is the case of two adjacent commercial trees, one of which is malformed and therefore selected for thinning. The other is the case of the large non-commercial tree which may be left unthinned because it is not directly competing with a commercial species.

4.6 **Case Study: The CAFOGROM cohort model**

4.6.1 *Background*

The CAFOGROM model was developed for the *Centro de Pesquisa Agroflorestal da Amazonia Oriental* (CPATU) by the author during 1994. It is a cohort model simulating the growth of a single stand, and is written in C. A demonstration version of the program can be ordered from the author as described in Appendix A. The C source code is listed in Appendix B. The general structure of the program has been described on page 97.

The analyses described in the following sections show how the functions for increment, mortality and recruitment were developed. Other aspects of the model, including the simulation of harvesting and thinning and the user interface, are described in section 5.

The database from which the model was developed comprises a series of 96 permanent plots arranged in 3 experimental dispositions in Tapajos Forest, Para, Brazil. This forest can be described as *terra firme* Amazonian tropical rainforest. Background information on the plots, their treatment, and the environment, are given in Silva (1989) and Silva *et al.* (1994). The plots have been measured since 1981, although some of the most recent measurements were not used in the analysis.

4.6.2 *Forming species groups for growth modelling*

Some 256 species are represented in the Tapajos data set. Many of these have only a small number of observations. The species therefore had to be grouped in some logical manner in order to ensure that each group was represented by sufficient data to fit the required growth models. Grouping by ecological and commercial groups represents one possible strategy. However, examination of the data suggested that existing groups encompassed a wide variation in growth rate and tree size, and were not directly suitable for growth modelling. This confirms the

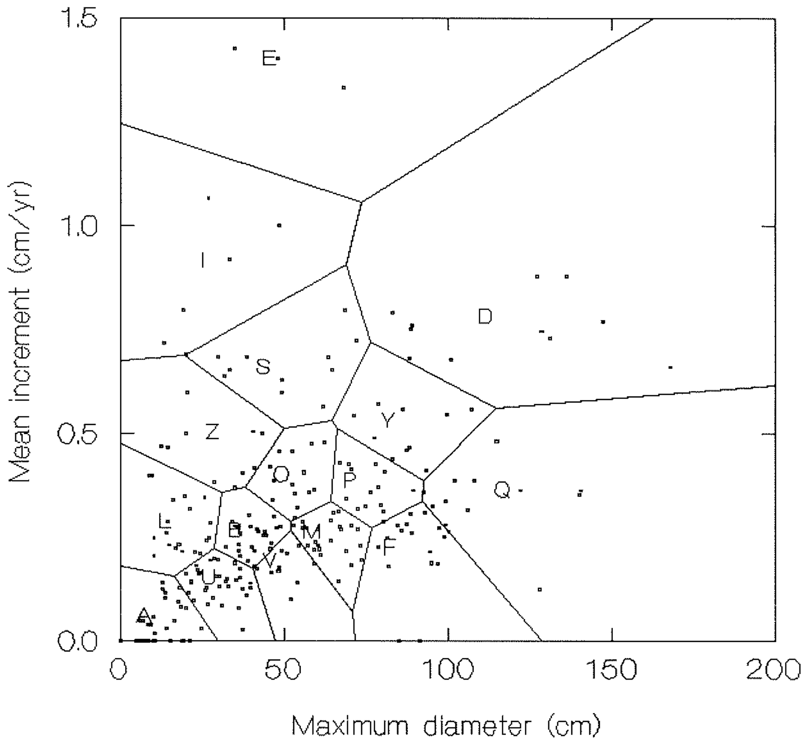


Figure 36 *Species clustered by mean increment and largest diameter into 16 groups for the CAFOGROM model*

author's experience with species in West African forests, which show a similar lack of correspondance between ecological guild and growth rate characteristics.

The method of grouping adopted was to use cluster analysis based on the K-Means strategy (Wilkinson, 1988). This splits a multi-dimensional data set into a pre-specified number of groups in such a way as to minimize within group variance, and maximize between group variance. Vanclay (1994:127) discusses cluster analysis and other strategies for grouping species regressions.

Two statistics were used as the basis for clustering: Mean diameter increment (DINC), and the diameter of the largest tree observed (DMAX). Initially, 26 groups were specified (to use the letters A-Z). However, 10 of the groups formed in this way had few trees in them*, and were amalgamated with the closest well-represented groups, giving a final assortment of 16 groups. The Voronoi diagram in Figure 36 shows how the groups are formed. Each tessellation shows the range of species-mean characteristics for a given growth model. The points show the locations of the individual species means. Thus, for example, model D comprises large, fast growing trees. Model A comprises very small and slow growing species. Model Q includes large, slow growing species. Model E are very fast growing trees which however attain only a limited maximum size of around 50 cm dbh.

The relationship between growth model groups and the commercial and

* The criteria used was to amalgamate all groups with less than 100 trees in them.

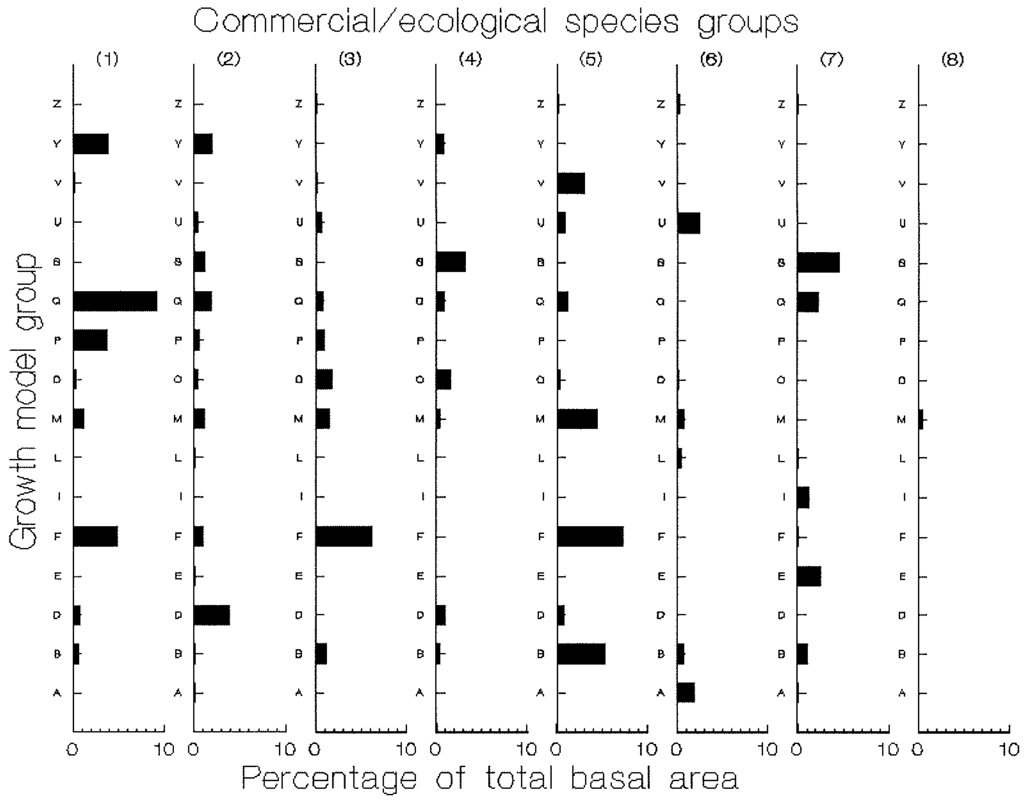


Figure 37 *Correspondence between growth models and commercial/ecological species groups in the data sets used to develop CAFOGROM*

ecological groups used by CPATU is shown in Figure 37. The numbering scheme of the commercial/ecological groups is shown in Table 11. In Figure 37 bars show the percentage of mean plot basal area contributed by a particular combination of growth model and commercial group. Thus for example, Q1 comprises about 10% of mean basal area on the 132 Tapajos plots analysed, and is the most important single combination, comprising (by reference to Figure 36 and Table 11) large, slow growing trees which are commercial, shade tolerant species.

The correspondence between the ecological guilds (shade tolerant and intolerant, pioneers) and the growth model groups can be seen in Figure 37, although it is not a completely exclusive relationship. For example, model F is prominent in the shade tolerant groups. Groups E and I, which have, as will be seen, extreme growth and mortality models, are characteristic of the fast growing, short-lived pioneer species. Groups S is confined to the light-demanding (shade intolerant) and pioneer groups. Model M, on the other hand, occurs in all groups except the pioneers (7), and is characteristic of unidentified species in group (8).

4.6.3 *Diameter increment regressions*

For the purposes of analysis, the data were grouped into size classes formed by $\log_2$ intervals of basal area for trees above 5 cm. This method of classification tends to equalize the number of trees in each class, and results in a reasonable number of data points for regression. Using conventional fixed-width classes tends to result in a very large amount of data in the first class, and a very unequal weighting for the regression.

Data was also classified by canopy class. Two shade classes were distinguished: Canopy trees, including emergents and co-dominants, and understorey trees. Canopy trees were defined as having crown illumination class (ILC) codes 1 or 2. Understorey trees were defined as ILC codes 3 or 0. This includes all trees less than 10 cm on which ILC was not assessed on the PSPs, some of which may be growing in gaps.

Commercial species:	
1	Shade tolerant
2	Shade intolerant
Potentially commercial species:	
3	Shade tolerant
4	Shade intolerant
Non-commercial species:	
5	Shade tolerant, large trees
6	Shade tolerant, small trees
7	Pioneer species
8	Species of unknown ecology
9	Palms

Table 11 *Species groups used in CAFOGROM*

Figure 38 shows the appearance of the data for four of the sixteen model groups. Each point is a class mean for a number of trees, with the error bar showing the standard error of the mean. The scales are logarithmic, with tree basal area in cm^2 on the horizontal axis, and tree basal area increment in cm^2/yr on the vertical axis. The solid lines are for canopy trees, and the broken lines for understorey trees. It can be seen that the pattern is approximately linear, although with a tendency to curve downwards in the larger sizes. There is a clear separation between canopy and understorey trees, with the latter tending to have lower growth rates in a given size class.

The following regression model was fitted to these transformed data:

$$\log(TBAI) = \alpha_0 + \alpha_1.CP + \beta_1.\log(TBA) + \beta_2.TBA \qquad \{67\}$$

This is based on the power-exponential model (see equation {39}, page 114) but with the CP term allowing for parallel curves for different shade classes. CP was coded as 0 for understorey trees, or 1 for canopy trees. The model was fitted to the 16 model groups, with the significance of each term being tested by stepwise regression. The regression was weighted by the square root of the number of observations in each class[*].

For all models except E and I, R^2 for the grouped, weighted, data was better than 90%, and better than 95% except for model A. Models E and I showed little change in mean increment with tree size, and the β_1 term was not significant. However, there were significant differences in increment with shade class, reflected in the α_1 coefficient. The relative difference in growth between shade and canopy trees is given by $\exp(\alpha_1)$, and range from 35% to 70%, in terms of basal area increment.

[*] A weakness of SYSTAT 5 is that weights must be integer values. Weighting by inverse of the variance, which is the most correct procedure, requires some investigation to determine appropriate scaling. The use of $\sqrt{N}$ gives a quick approximation that avoids giving undue weight to sparse observations, as would an unweighted analysis.

Model	No. Trees	No. Species	α_1	α_2	β_1	β_2
A	1734	39	-2.5790	0.1491	0.6795	-0.000802
B	3402	26	-1.3100	0.2937	0.5467	
D	563	12	-2.0970	0.5296	0.8976	-0.000184
E	425	3	2.8720	0.4295		0.000531
F	4774	14	-2.0040	0.2445	0.7138	-0.000115
I	249	5	2.4660	0.4992		
L	229	23	-1.1110	0.4372	0.5054	
M	2311	24	-2.3430	0.3542	0.7494	-0.000139
O	875	12	-2.1350	0.5080	0.8455	-0.000764
P	925	12	-1.6955	0.4475	0.7102	-0.000148
Q	2259	19	-1.3030	0.3393	0.5970	
S	2916	8	-0.7760	0.3420	0.6638	-0.000154
U	2889	27	-2.3206	0.3455	0.6812	
V	1306	10	-2.3446	0.2471	0.7296	-0.000121
Y	694	9	-1.4260	0.4560	0.6959	-0.000164
Z	160	13	-2.1807	0.3403	0.8973	-0.000915

Table 12 *Coefficients for the CAFOGROM basal area increment model for each species group*

It is easier generally to visualize and work in terms of diameter increment. The relation between basal area and diameter increment is given in equation {37} on page 107. Equation {67} can thus be transformed to give:

$$\mathrm{DINC} = \exp[\alpha_0 + \alpha_1.\mathrm{CP} + \beta_1.\log(\tfrac{1}{4}\pi.\mathrm{DIAM}^2) + \beta_2.(\tfrac{1}{4}\pi.\mathrm{DIAM}^2)] \div (\tfrac{1}{2}\pi.\mathrm{DIAM}) \qquad \{68\}$$

This equation is plotted in Figure 39 for two extreme examples, models A and D, to show the typical shape of the model in terms of diameter increment and diameter. As before, solid lines and triangles depict canopy trees. Broken lines and open triangles are understorey trees. It will be noted that the vertical and horizontal scales differ between the two graphs. Reference to Figure 36 will show that model A relates to small, slow growing trees, and D to large, fast growing ones.

4.6.4 *Predicting the canopy status of trees*

The diameter increment model uses canopy position (CP) as one of its predictor variables. This implies that in a dynamic model, it must be possible to predict changes in the CP status of trees as simulation proceeds. It is not possible to determine dynamic functions for individual tree crown status directly from PSPs, because canopy status changes only very slowly over time. Measurements over 5-10 year periods give very little information on such changes. However, it is possible to make assumptions based on static models to assess how many upper canopy trees may be present at a given time.

Figure 40 shows how the proportion of plot basal area comprised of trees in the upper canopy changes with increasing total plot basal area. As basal area increases, the proportion of total basal area in canopy trees declines. The data is quite variable, with the fitted regression being:

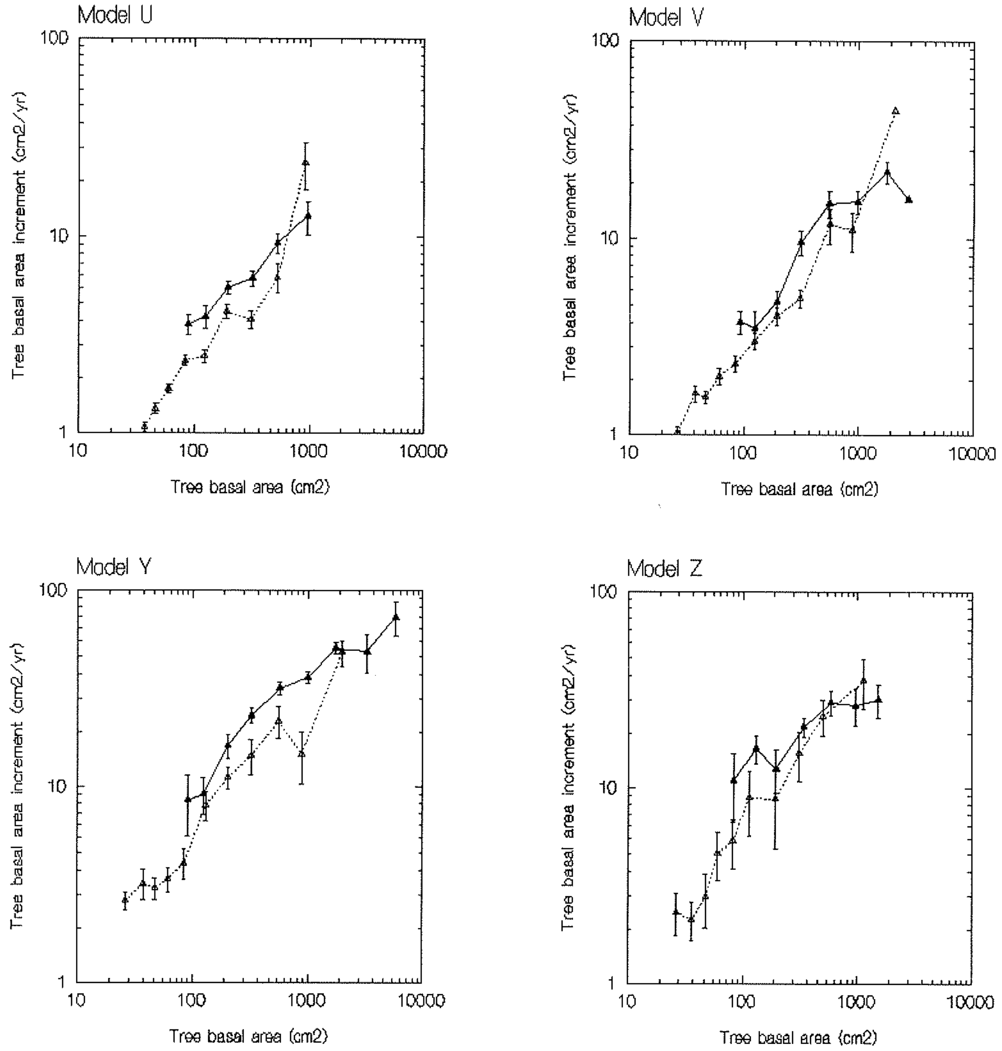


Figure 38 *Tree increment data used in CAFOGROM grouped by size classes, for 4 species groups.*

$$CTBAR = 0.80 - 0.0072\ SBA \qquad \{69\}$$

where CTBAR is the canopy tree basal area ratio, and SBA is stand basal area in m²/ha. With 132 plots, this regression had an R^2 of 0.289. Although this is rather low, the 95% confidence limits of the regression line are well-determined, as shown on the graph, due to the large number of points.

In the model, this regression can be used to adjust the numbers of canopy trees to or from the understorey class. If the basal area ratio exceeds that predicted by the regression, then there are too many upper canopy trees. The smallest can then be demoted into the understorey class until the level required by equation {5} is achieved. Conversely, if the ratio is less than that predicted, the largest understorey trees should be promoted.

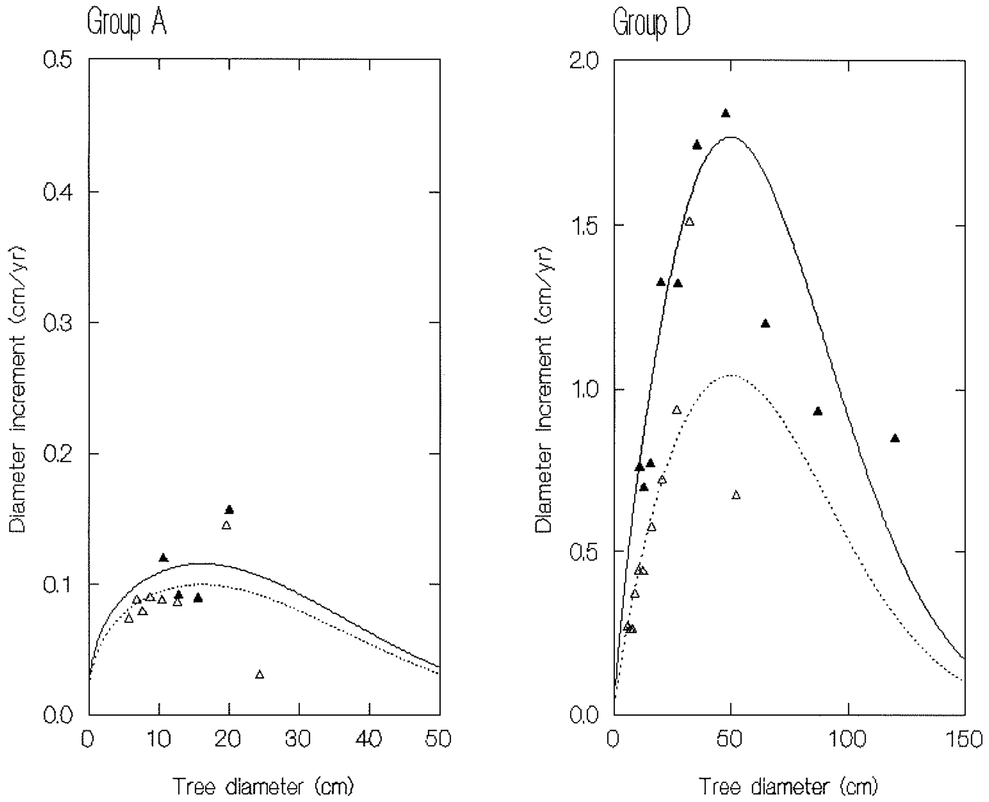


Figure 39 *Increment models A and D from CAFOGROM shown in terms of tree diameter and increment*

4.6.5 *Mortality, canopy status, and logging conditions*

Mortality trees were summarised by the model groups A-Z, and by classes of canopy status and logging condition. Logging conditions were coded from 1 to 4 to cover undisturbed forest, forest recovering more than 5-years after logging, lightly logged experimental plots, and the more heavily logged plots. The mortality data was transformed by a logistic transformation (see page 143, equation {63}) in order to normalize it for analysis of variance (ANOVA). An ANOVA was performed on the model:

$$\mathit{logit}(\mathrm{AMR}) = \alpha + \mu_{\mathrm{ci}} + \mu_{\mathrm{lj}} + \mu_{\mathrm{sk}} + \epsilon \qquad \{70\}$$

Here, μ_{ci}, μ_{lj}, μ_{sk} are the mean effects due to canopy position, logging history, and species model, respectively. α is a constant, and ϵ are the normally distributed errors for each observation. *Logit*(AMR) is the transformed value of the Annual Mortality Rates, using equation {63}.

The ANOVA showed that the heavily and lightly logged plots did not differ in their mortality rates significantly, whilst the recovering (more than 5 years after logging) forest was closely similar to the unlogged forest. It was accordingly decided to simplify the analysis by grouping the mortality data into four classes for each species. These comprised either understorey or canopy trees, combined with plots that were either immediately post-harvest or unlogged/recovering. In

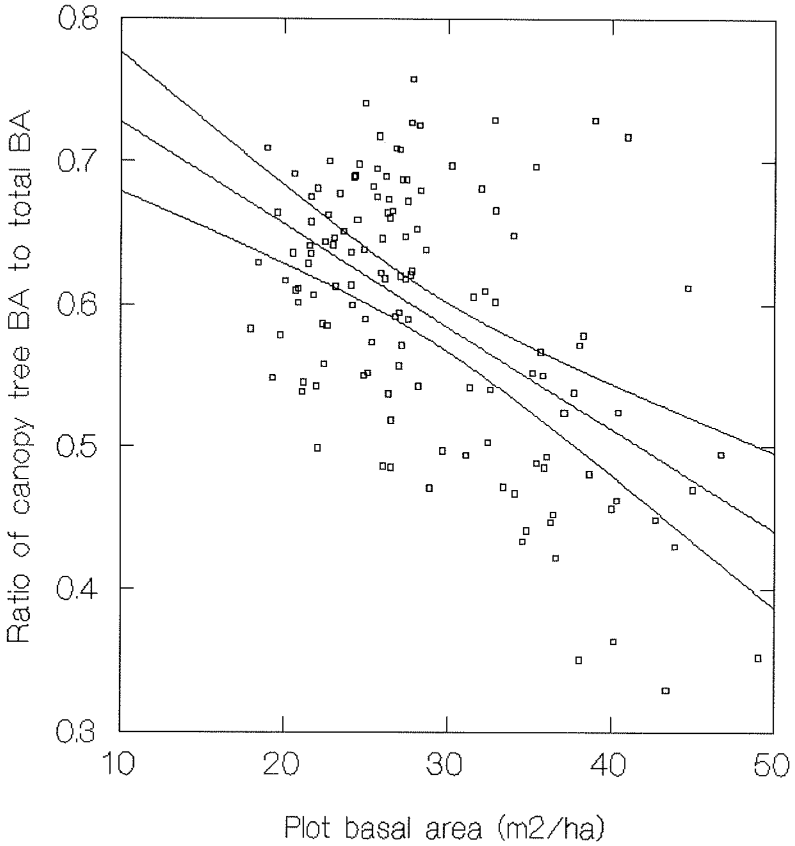


Figure 40 *Predictive model used in CAFOGROM to relate number and basal area of upper canopy trees*

this way, each growth model is represented by the four annual mortality rates, as shown in Table 13.

It may be noted that the figures give logical and consistent indications when compared with the defining model parameters in Figure 36. Thus for example, models E and I show typical extreme pioneer behaviour, with very high mortality as understorey trees in unlogged forest, and high mortality even as canopy trees; but with much lower death rates in the logged forest. Model Q is somewhat similar but less extreme. Conversely, models A and P characterise shade tolerant groups, with low mortalities in undisturbed forest in both shade and canopy, and much higher mortality in both categories after logging. Various other groups show intermediate behaviour of both types.

Growth model group	Unlogged		Logged		Dmax
	U/S %/yr	Canopy %/yr	U/S %/yr	Canopy %/yr	(cm)
A	2.42	3.68	2.75	3.50	24.5
B	2.81	1.68	2.38	2.52	47.5
D	2.86	2.28	3.22	2.97	131.3
E	16.33	2.71	2.67	1.12	68.0
F	1.58	1.50	1.57	1.62	94.7
I	15.43	3.98	8.27	1.44	48.4
L	2.43	5.41	1.96	2.38	27.3
M	1.89	1.10	1.63	1.63	73.4
O	2.98	1.69	2.75	2.11	62.0
P	1.41	1.01	1.04	1.63	79.4
Q	7.62	2.11	4.11	1.29	140.5
S	4.83	2.57	3.29	3.02	64.5
U	1.72	2.32	2.21	2.35	39.6
V	1.02	1.33	1.80	1.27	58.8
Y	2.80	1.05	1.39	1.13	107.0
Z	5.17	2.47	2.65	1.17	48.4

Table 13 *Annual mortality rates (%) and largest tree diameters for each growth model group in CAFOGROM*

It was found in validation of the model that small proportions of the stocking of each model group would survive and grow past their observed maximum diameters (Dmax in Table 13). This was especially noticeable for the extreme pioneer species groups E and I. Reference to Table 12 and equation {67} will show that for these, there is no downward trend observed in the data for growth rates of larger trees (negative β_2 coefficient). To eliminate this unrealistic feature and promote greater accuracy, the mortality model was modified so that any trees which had grown to greater than the observed maximum diameter were assumed to die within a given 5-year period.

4.6.6 *Recruitment models*

The approach to modelling recruitment involves two parts. The first is to predict the total number of recruits as a function of the basal area lost during a growth period. The second involves apportioning this recruitment to pioneers, light demanders (shade intolerant species) and shade bearers (shade tolerant species).

Figure 41 shows how the total level of recruitment was modelled. Plotting number of ingrowth trees (NIN) over a 5-year period on stand basal area at the start of the regression gave a statistically significant but rather weak regression. Validation tests with the complete model showed that net basal area growth was sensitive to recruit levels following logging, and that the response was quite dynamic. The regression model did not reflect this well, whereas a principal components model gave the line shown in the graph. The ellipsoid shows the 50% contour of the joint normal distribution of basal area and numbers of recruits. The line is drawn through the principle axis of the ellipse. Estimating the parameters graphically gave a model:

$$\text{NIN} = 1526 - 52.6\ \text{SBA} \qquad \{71\}$$

Neither the methodology nor the model are ideal. This question requires further and more detailed analysis, but it proved adequate to give reasonable results for a provisional model. Equation {70}, {71} will predict zero recruits at a basal area of 29 m²/ha. In the model, this was not allowed, and a small number of recruits (50/ha/5-year period) always occur irrespective of total basal area. An exponential type model would give this behaviour naturally and would be a better choice of function.

Having determined a model to estimate the total number of recruits, it was then necessary to consider how these could be partitioned among species. After several trial analyses with different options, the best results were obtained with the functions shown in Figure 42. These use the total number of recruits (NIN) as an indicator of disturbance, with each regression predicting the levels of pioneer, shade tolerant, and shade intolerant species respectively. The general form of the regressions was:

$$\mathit{logit}(\text{P}) = \alpha + \beta.\log_e(\text{NIN}) \qquad \{72\}$$

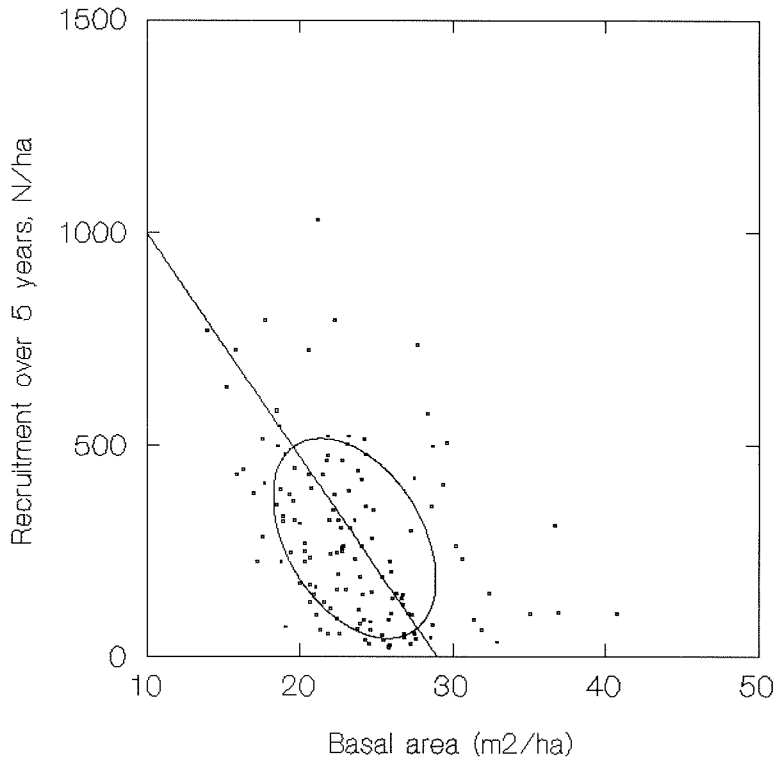


Figure 41 *Function relating number of recruits to basal areas losses in CAFOGROM*

where *logit*(P) is the logistic transformation (see equation {63}) of the proportion of recruits belonging to the specified guild. The central graph shows two curves. The dotted line is the curve fitted by regression, whilst the solid line is a calculated curve derived from the regressions for pioneers and shade intolerants in order to force the total proportions to sum to 100%. It is this calculated curve that is finally used in the model.

Transforming equation {72} to give P directly, the three predictive models were therefore:

(i) Pioneers (Group 7):

$$P_p = 100 \div [\,1 + 1 \div (0.0431\ \mathrm{NIN}^{0.456})] \qquad \{73\}$$

(ii) Shade tolerant (Groups 1,3,5,6):

$$P_t = 100 - (P_p + P_i) \qquad \{74\}$$

(iii) Shade intolerant (Groups 2,4):

$$P_i = 100 \div [\,1 + 1 \div (0.0529\ \mathrm{NIN}^{0.309})] \qquad \{75\}$$

It can be seen that the equation for group (ii) is formulated as the difference between the proportions of the other two species, so that the three ecological

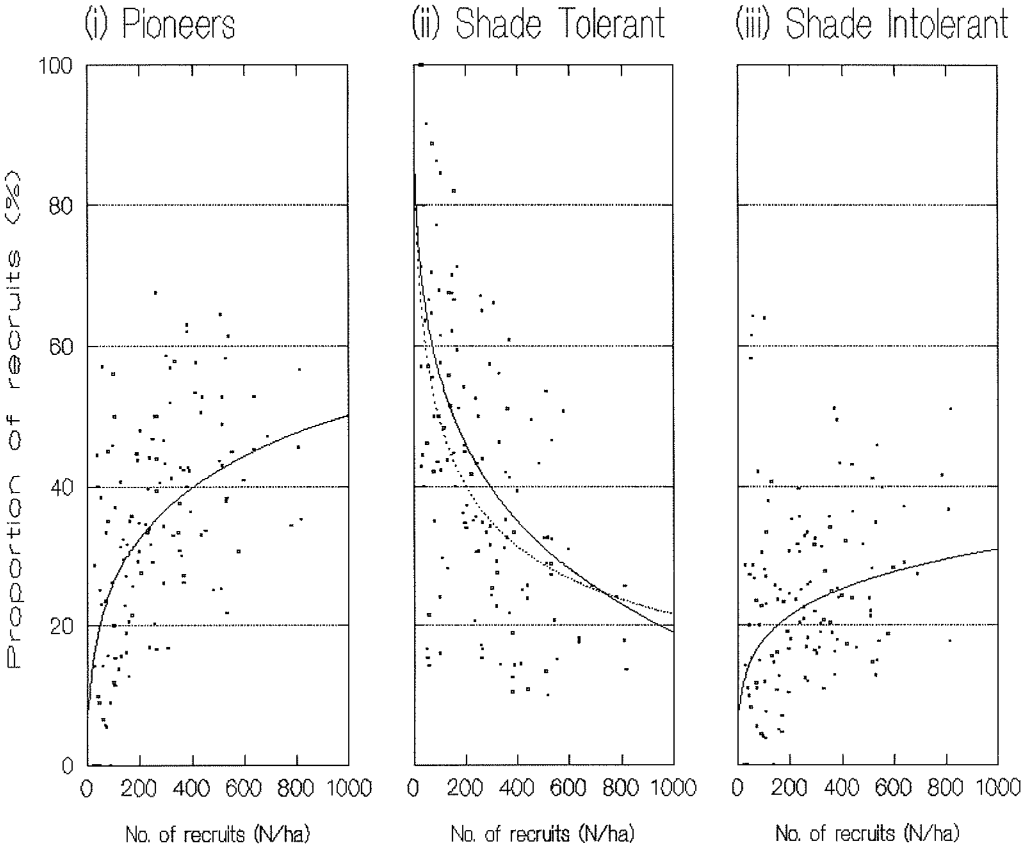


Figure 42 *Proportions of ecological types in recruitment depending on disturbance*

groups will add to 100%. The equations for pioneers and shade intolerants are fitted by regression.

Examining Figure 42, it can be seen that, as may be expected, with increasing disturbance, the proportion of pioneer species rises steeply, but approaches an asymptote of perhaps 55-60%. The proportion of shade tolerant species is very high in lightly disturbed stands, but falls steeply to below 20% as disturbance increases. The shade intolerant group follows a similar but lower curve to the pioneers and is never a major component of the regeneration.

In order to translate these proportions for ecological guilds into specific numbers of recruits for particular species groups, tables of recruitment proportions were prepared for each of the major types. The unclassified species (group 8) were included with shade bearers. The tables are made for each combination of species group (1-8) and growth model (A-Z), but list only those which have non-zero occurrence.

Initially it was assumed that recruits began a particular growth period at the minimum measurement diameter of 5 cm. However, during validation it was found that recruitment basal area was being significantly under-estimated. Calculating mean diameters of recruits for each growth model group corrected this problem. The mean recruit diameters are shown in Table 14. It will be seen that in particular the pioneer groups E and I (mainly comprising *Cecropia* species) have quite large initial diameters, indicative of their fast early growth. The rapid recovery of basal area following logging is largely attributable to the occurrence of a large number of recruits of these species. This aspect of stand dynamics is well illustrated by the model.

Model	Diam (cm)
A	6.14
B	6.07
D	6.48
E	13.80
F	6.04
I	10.58
L	6.04
M	6.34
O	6.33
P	6.30
Q	6.82
S	7.71
U	5.83
V	5.94
Y	7.05
Z	6.68

Table 14 *Mean diameter of recruit trees in CAFOGROM, by growth model group*

4.6.7 *Logging damage: Gaps, skid trails and landings*

One of the major impacts of logging is the destruction of forest by tractors and debris during felling, skidding and loading. In CAFOGROM, data are used from Silva et al. (1994) which indicate that with the removal of 70 m^3/ha, about one-third of the standing volume, some 48% of the forest is destroyed as skid trails, felling gaps, and landings. This led to a simple *ad hoc* function that estimated the area lost through logging damage as:

$$AL = (HVOL/TVOL)^{0.575} \qquad \{76\}$$

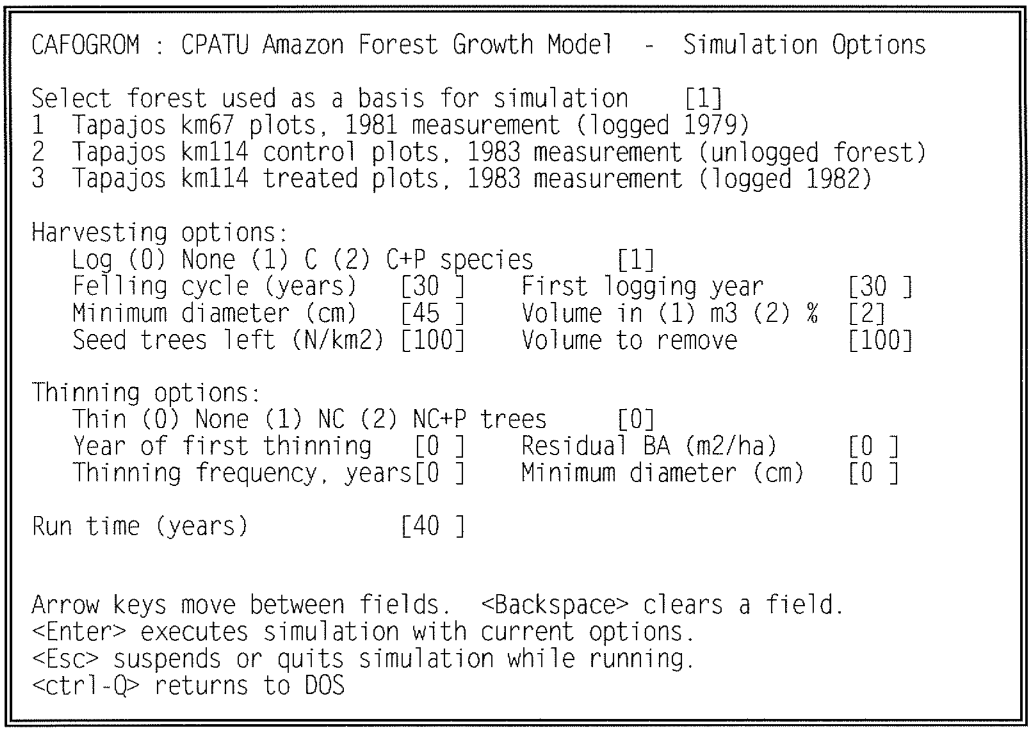
where AL is the area lost as a fraction of the total, HVOL is the volume harvested, in m^3, and TVOL is the total volume standing before logging. This function gives 50% losses when the logging intensity (HVOL/TVOL) is 30%.

4.6.8 *The CAFOGROM user interface*

The CAFOGROM program communicates with the user via two basic screens. The first, shown in Figure 43, allows the user to set parameters for a given run. These cover the definition of the data set to be simulated, the harvesting and thinning options, and the total time for the simulation. Lines 84-250 of the CAFOGROM listing are responsible for setting up this screen and passing the parameters set by the user back to the program.

The second screen (Figure 44) is displayed whilst the simulation is running, and shows the output from the model. It is a graphical display with three regions:

(i) *Basal areas by diameter classes*: At the top of the screen, basal areas are displayed by diameter classes for each period of the simulation. The leftmost bar is for trees 5-25 cm, the next for 25-45 cm, and so on. These classes were chosen to reflect both the 5-cm minimum diameter recognised in the model, and the 45-cm minimum felling diameter commonly adopted in Brazil. The bars are sub-divided by ecological categories, into pioneer species (black), shade intolerant non-pioneers (shaded) and shade tolerant species (white).



```
CAFOGROM : CPATU Amazon Forest Growth Model  -  Simulation Options

Select forest used as a basis for simulation    [1]
1  Tapajos km67 plots, 1981 measurement (logged 1979)
2  Tapajos km114 control plots, 1983 measurement (unlogged forest)
3  Tapajos km114 treated plots, 1983 measurement (logged 1982)

Harvesting options:
   Log (0) None (1) C (2) C+P species      [1]
   Felling cycle (years)    [30 ]    First logging year      [30 ]
   Minimum diameter (cm)    [45 ]    Volume in (1) m3 (2) %  [2]
   Seed trees left (N/km2) [100]    Volume to remove        [100]

Thinning options:
   Thin (0) None (1) NC (2) NC+P trees     [0]
   Year of first thinning   [0 ]    Residual BA (m2/ha)     [0 ]
   Thinning frequency, years[0 ]    Minimum diameter (cm)   [0 ]

Run time (years)          [40 ]


Arrow keys move between fields.  <Backspace> clears a field.
<Enter> executes simulation with current options.
<Esc> suspends or quits simulation while running.
<ctrl-Q> returns to DOS
```

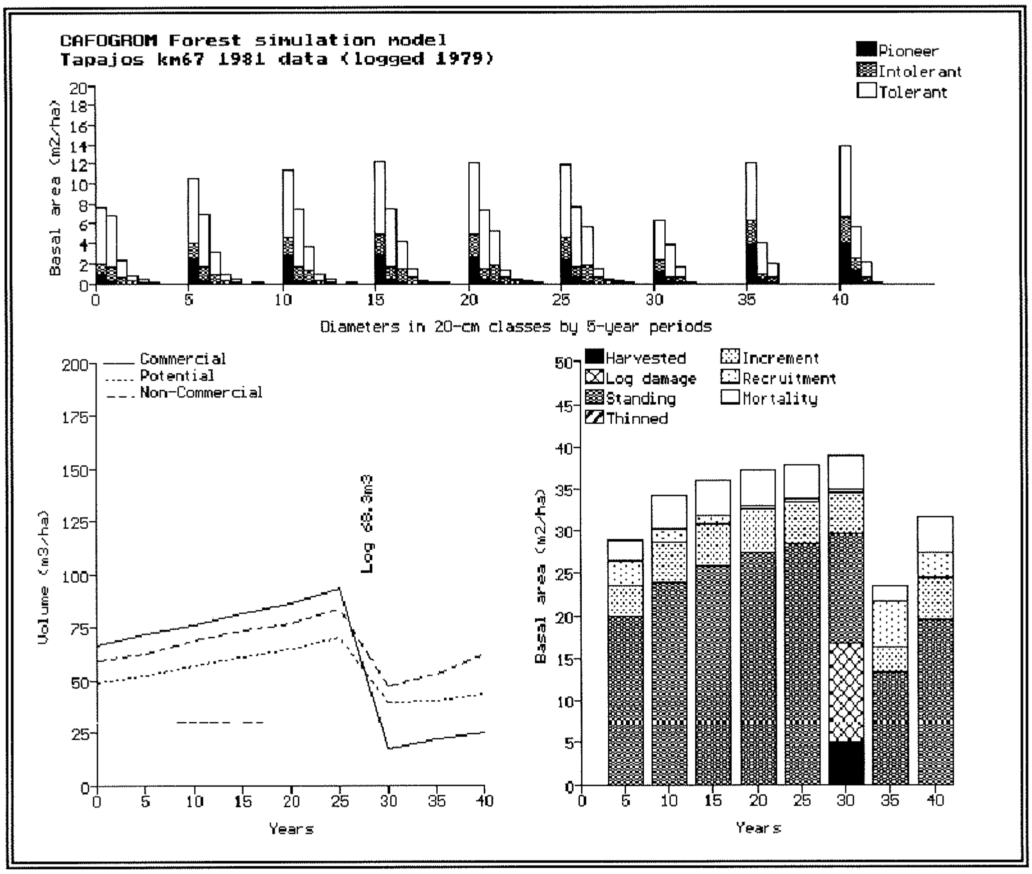
Figure 43 *The CAFOGROM input screen to set forest management parameters*

(ii) *Volume over time*: This graph appears on the lower left part of the screen. It is probably the output of most interest to the forest manager. It shows the standing bole volume for all trees over 15 cm dbh for three species groups: Commercial, potentially commercial, and non-commercial species. Volumes removed in fellings or basal areas thinned are shown as annotations on the graph.*

(iii) *Basal area components*: At the lower right of the screen is a graph showing the components of basal area (BA). This is instructive in understanding the dynamics of the natural forest. The main shaded area is standing BA at the start of a 5-year simulation period. Above this is a close-spaced stipple representing increment on the standing trees.

A wider-spaced stipple then shows the increment accruing over a 5-year period on recruits into the 5-cm or higher size classes. The white, unshaded part of the bar shows losses from mortality. The black part of the bar appears when harvesting occurs, and shows the part of the standing basal area removed in logging as extracted trees. Above this is a cross-hatched component which shows the basal area destroyed by felling gaps, skid trails and roads.

* The small horizontal line appearing at about 25 m^3 is an artefact of the frame grabber used to capture the image for the word processor, and does not appear on the computer screen.



CAFOGROM Forest simulation model
Tapajos km67 1981 data (logged 1979)

Figure 44 *CAFOGROM output screen with graphs of simulation results*

5 GROWTH MODEL VALIDATION AND APPLICATION

5.1 Growth model validation

5.1.1 *Definition and role of validation*

The validation of a model or computer program is the process of checking inputs and corresponding outputs to determine accuracy. In the case of a forest growth model, this means comparing the projected growth of stands over various periods of time and under various management regimes with the observed growth, as determined from PSPs, experimental plots, and stands of known age since logging.

The data used to test a model may be completely independent of the data used to construct it, or it may be the same data. These two situations can be referred to as *independent* or *self-validation*. Self-validation is a normal procedure with simple regression models, where statistics such as R^2 or residual standard deviation provide indicators of regression accuracy. Residual plots are used in regression analysis to determine undesirable features such as lack of fit, heteroscedacity, or bias (Draper & Smith, 1966:83). Independent validation is more rigorous than self-validation, but when data is limited, may be an unachievable ideal. As an example, Alder (1979) used both self- and independent validation to test the VYTL growth model. Self-validation was based on the same PSPs as were used to develop the model. Independent validation was performed by simulating a number of thinning experiments whose data had not been used in the model.

It may be the case that no suitable data exists to fully validate a model. This case arose, for example, with the GHAFOSIM study (section 3.2.3, page 88). GHAFOSIM was based on individual tree growth data, from which whole stand yields were estimated. There was no data available on whole stand growth that could be used to check the model. Validation in such cases is limited to determining that model outputs are reasonable, conforming to general expectations and published results from similar forest types.

Formal documentation of the validation process is important to the acceptance of a model as a management tool. Introduction of growth modelling techniques into forest management implies and requires the evolution of a forest management information system. This in turn imposes a need for organizational change. Such changes will tend to be resisted by individuals who are threatened or challenged by them. If no formal validation has been performed, it will be relatively easy to oppose the introduction of modelling techniques on the grounds that the model is unproven and of doubtful accuracy.

Conversely, there may be a tendency to an over-enthusiastic and uncritical adoption of computer methods without thorough testing. In such cases, programming errors ('bugs') and model weaknesses could lead to grossly erroneous management decisions. Apart from the direct environmental and fiscal damage that might ensue, this would have the effect of discrediting modelling techniques, and retarding their wise and proper use.

Although a model should be formally validated as a terminal component of its development and documentation, validation is also an ongoing process during its composition. It is applied to check individual growth functions and also complete program modules. This internal validation will generate changes and adjustments to functions used, especially with respect to bias. Validation thus becomes a component of function and program development.

5.1.2 *Types of error in growth models*

The types of error that will be detected during validation, and the corresponding remedial measures to be taken, will be of several kinds:

(i) *Unexplained variation*: The correlation of dependent with predictor variables may be very low. This will be indicated by a low R^2 to a regression. An R^2 of less than 50% indicates a function of little real use: it could equally be replaced by a simple average, and a stochastic modelling approach may be better indicated than one using regression models.

(ii) *Lack of fit:* Residual plots may show a significant regression with test variables, indicating that the model could be improved through the addition of further terms. Alternative regression functions need to be tried to reduce the lack of fit.

(iii) *Bias:* Residual plots may show a distinct tendency for points to be above or below the zero line. This can be compensated for by adding terms to a model to remove the bias. Bias commonly arises in regression models as a result of transformation of the dependent variable. When the fitted Y is transformed back to its original form, the result will be a biased function. Meyer (1944) suggested a correction factor for logarithmic transformations. General correction factors can be derived by fitting a regression to the residuals.

A small bias in increment, mortality or recruitment functions will tend to accrue as a large bias over time in a growth model, as the function results are added to earlier ones with each successive time increment. It is thus quite possible that bias will appear negligible when individual functions are fitted, but may yet appear substantial and unacceptable during validation of the complete model.

(iv) *Functional pathologies:* Some fitted functions may be termed pathological, in that they produce extremely odd results when used in a simulation model in spite of having high R^2 and apparently well-behaved residuals. An example is the modified Beta function shown in Figure 20 on page 117. Causes of such behaviour are:

- *Multiple inflection points.* High-order polynomials, nonlinear functions with several power coefficients, and functions of containing trigonometric terms such as $\alpha.\sin(x-\beta)$ can have many

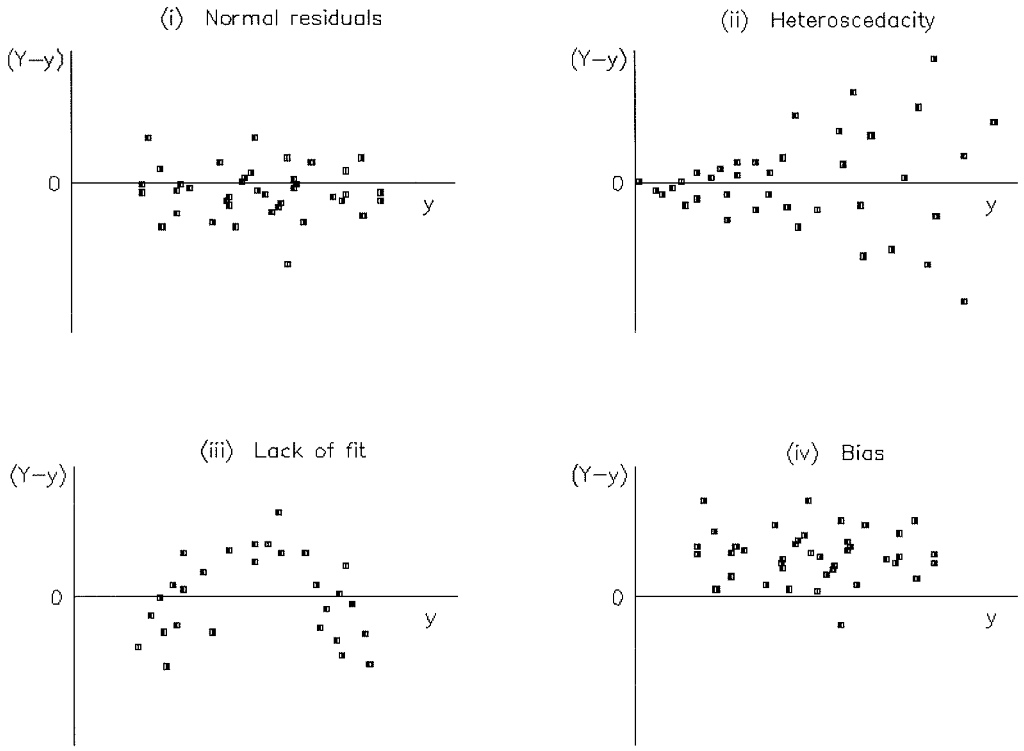


Figure 45 *Residual plots showing heteroscedacity, lack of fit, and bias*

inflections, and consequently fit scattered data quite well, whilst yet being completely meaningless as projective models (Figure 46).

- *Singularities:* Functions incorporating real powers, logarithms or exponents will tend to infinity or produce imaginary results outside a certain range of values. This may either cause a model to 'crash' or produce meaningless results, depending on individual features of the programming environment (*eg.* Figure 20, page 117).

- *Covariances among predictor variables:* If a regression incorporates several terms which are closely correlated, the effect is similar to fitting a higher-order polynomial. The behaviour of such equations is quite unstable, and depends on the particular correspondences that exist among the predictor variables in the data set.

(v) *Logical and programming errors*: As any programmer will know, a working computer program is no guarantee that a computer program works. There are many kinds of logical errors that can arise including incorrect mathematics, typing mistakes leading to the use of a wrong variable name, and unforeseen combinations of input parameters. A program may appear to function perfectly correctly, yet may contain subtle mistakes that are not easily noticed. Errors of this type can be minimized by modular program design, with each module being completely and thoroughly tested. A desirable target is to limit individual procedures to

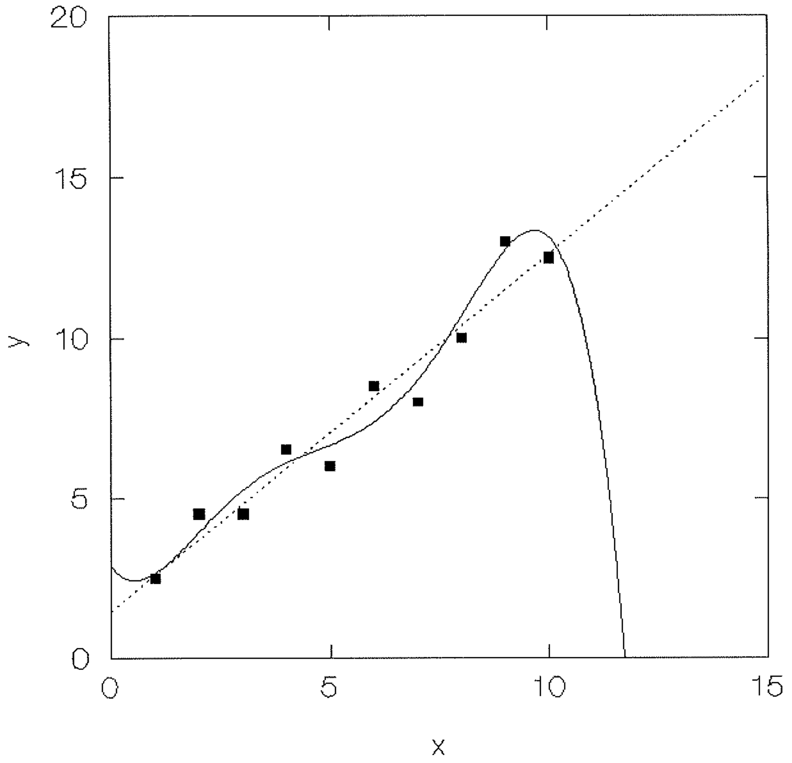


Figure 46 *A pathological function : The fifth-order polynomial fits the data well but behaves erratically when extrapolated over a short distance*

a maximum of 50 lines of code, and to test each procedure independently. Structured programming, in which all modules receive and return their results as parameters, is also good practice, but may be difficult to apply with a database language, in which field names are implicitly global variables.

In principle, all the above sources of error except (i) can be controlled and minimized through an ongoing validation process. The problem of unexplained variation is, ultimately, a problem of research, experimental design, and data acquisition.

5.1.3 *Residual analysis and correction for bias or lack of fit*

Residual analysis is a useful formal tool for validating models. It tends to be especially applied to individual regression functions, but it is equally useful applied to composite processes such a complete simulation model.

The residual is defined as:

$$E = Y - y$$

where:

E is the residual,
Y is an observed value,
y is the equivalent predicted value from the model.

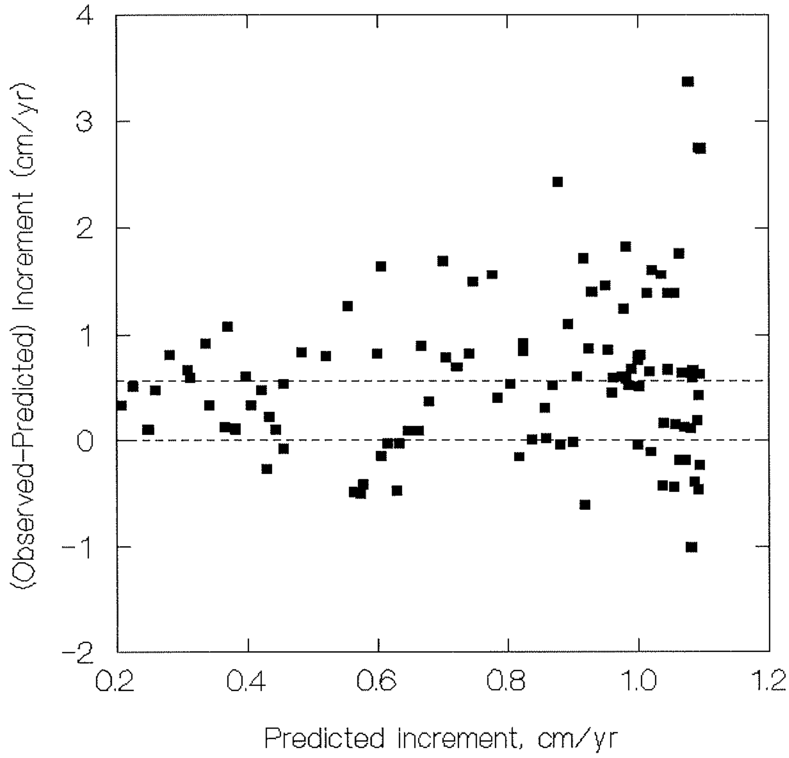


Figure 47 *Residuals of diameter increment from equation {46} applied to a set of independent test data*

A predictive system should ideally give unbiased residuals with no consistent trends left unaccounted for by the dependent variables of the model. Bias can be detected if the sum of the residuals differs significantly from zero. Residual trends, or lack of fit, can be determined by the Wilcoxon rank sign test. Calculation details for this test are given in Philip (1994); it will also be available in most statistical packages.

For the underlying assumptions of regression analysis to be fully respected, residuals should also show homogenous variance. That is, the variance of the residuals should be uncorrelated with the predicted value.

Residuals can be examined graphically by plotting them against the predicted value, or against one of the independent variables of the model. The former is the most useful test of general model performance. When particular problems are noted, they can be analysed in more detail by making further plots of residuals against selected independent variables. Figure 45 shows the types of residual plots that may be obtained. Normal residuals, unbiased, and with no systematic trends, are shown in (i). This is the ideal result. Heteroscedacity, or non-homogenous variance, is shown in (ii). Lack of fit is shown in (iii). This is denoted by runs of positive or negative residuals when they are ordered by predicted value. Bias is shown in (iv).

In regression analysis, residuals are plotted directly using values from the linear model, and should automatically be unbiased unless the conventional statistical model has been modified in some way. Texts on regression analysis such as Draper & Smith (1966) or Seber (1977) consider aspects of residual analysis in detail.

However, in mathematical modelling, the important residual is that which arises when the function is used as a predictive system. In most cases, the function will be designed in nonlinear form. It will then be transformed into a linear equivalent for parameter estimation, and will be back-transformed in order to be used. In simulation modelling, predictions of events over short intervals of time will be summed or numerically integrated to give an aggregate result at the end of the prediction period.

Thus the unbiased, linear regression model, ideally with homogeneous variance, will be subject to back-transformation and numerical integration in use. This will inevitably introduce a degree of bias and lack of fit. The extent of these defects will be determined by validation. With very simple models, a correction factor may be either estimated from the residuals, or determined analytically. Meyer (1944) showed, for example, that when a logarithmic function was used to fit a volume equation, the result was biased. This bias could be corrected with a formula based on the standard deviation of the residuals from the regression (for details, see Philip, 1994, §2.7.4).

These points are illustrated in the residual plot shown in Figure 47. This applies equation {46} on page 119 to a set of independent test data. The resulting plot shows both bias and heteroscedacity, but no significant lack of fit. The dotted lines on the graph indicate the ideal, zero residual, and the actual mean residual of 0.57. The latter indicates that the function would, for the test data set, underestimate increment on average by 0.57 cm/yr. In this case, the explanation for the bias may be that the original regression data was drawn from selection forest, whereas the test data had been managed under the tropical shelterwood system. The latter typically results in significantly higher increments (Osafo, 1970; Alder, 1993) due to the promotion of juvenile stems and the greater crown freedom. The validation would indicate that this model would underestimate the growth response of stands to liberation treatments.

It may be noted that a mean negative residual indicates *overestimation* by the model, or positive bias. Conversely, mean positive residuals indicate underestimation by the model or negative bias.

Heteroscedacity is not an important defect. Its relevance is mainly to regression analysis, where it indicates that the dependent variable should be transformed in order to obtain the maximum liklihood estimate of parameters using conventional least squares methods.

An example of residual analysis applied to a complete forest simulation model is given in Alder (1979), in an application involving tropical plantation conifers. The growth model was embedded in a test program that simulated each PSP in the data set in turn, and compared actual and observed total volumes. These were summarised to tabulate mean bias by species and length of projection period. Over a wide range of model conditions, bias in volume estimation was less than 5%, whilst the standard deviation of the residuals ranged from 10-20% of predicted volume. This was an example of self-validation, in that the test was carried out on the data set used to develop the growth model functions. It showed that the effects of function transformation and integration were within acceptable limits.

5.1.4 *Function plots and overlays*

Residual analysis is an important tool but it may be inappropriate or impossible in some instances. It is also, on its own, not sufficient to give complete confidence in a model. An additional requirement for a comprehensive validation study is to perform function plots, showing how the model performs over long periods of time and with extreme conditions of input parameters. These plots should ideally compare predicted and actual sets of data. Graphs of this kind are easily interpreted by forest managers and are less abstract than residual analysis.

Residual analysis of the complete model may not be possible when the model is built from basic data about individual tree growth or dimensions, or from a basis of theoretical process modelling, and there is no whole stand comparison data. In such cases function plots of model performance can be compared with general published information about system behaviour without requiring detailed and specific measurements.

Function plots are also most important in building individual growth functions as may be noted by the many examples in this book. The basic sequence in developing a function involves:

- A graph plot of the data, with a smoothed line applied using a spline function or some similar method supplied by the statistical package in question.

- Consideration of an appropriate function shape to fit the model.

- Transformation of the data, and replotting on linear scales with the fitted regression line or surface.

- Replotting on standard scales with the back-transformed function.

Once a complete growth model has been developed, then function plots should be made of:

- Basal area development of stands over time, with and without logging. These should show asymtotic growth to a maximum basal area of the order of 40 m^2/ha, perhaps fluctuating about that level in the unlogged forest. The logged forest should show recovery to full basal area over a 15-40 year period, depending on site and logging intensity.

- Species mixtures over time. These should typically show a decline in pioneer species in unlogged forest, and a stable and higher level in logged forest. Heavier logging should promote more pioneer species. Selective logging should tend to favour non-commercial species unless there is a policy of retaining seed trees.

If time-series data of stand volumes for different periods after logging are available from forest inventories, then these can be overlaid as data points on these function plots.

Function plots should always be extrapolated outside the range of the range of the data set used to develop the model. In this way, catastrophic emergence of model pathologies may be detected which would be hidden on residual plots. Unexpected features of the model may appear in long-term projections, as in the convergence of stand structure to a stable state in Example 41, page 88 *ff*.

Where logging or treatment experiment data is available over long periods, or growth data from long-term PSPs, these can be individually simulated and compared with the model. Alder (1979) simulated two thinning experiments and compared them graphically with actual data. This type of plot shows very clearly the strengths and weaknesses of a forest growth model.

5.1.5 *Validation example: Behavioural characteristics of the CAFOGROM model*

CAFOGROM, as an example of a cohort model, has been discussed on page 165 and elsewhere in this manual. The model was validated during development using a test version written in Foxpro. The tabulated output of this model in terms of basal area components was compared with the original PSPs used in the data analysis. During this process several amendments were made to the growth functions to improve the performance of the model.

For example, it was found in an early version that cumulative mortality over 10 years in terms of basal area was over-estimated. When the data was examined in detail, a mistake was found in the way mortality had been calculated. It had been assumed that a dead tree at the end of a measurement period indicated mortality during that period. However, in fact, some trees were already reported dead at the start of the period. It was therefore necessary to adjust the logic of the program calculating mortality, and re-estimate the mortality coefficients.

However, no data was available to fully test the model over longer periods of time. To do this, reliance was placed on examining the general behaviour of the

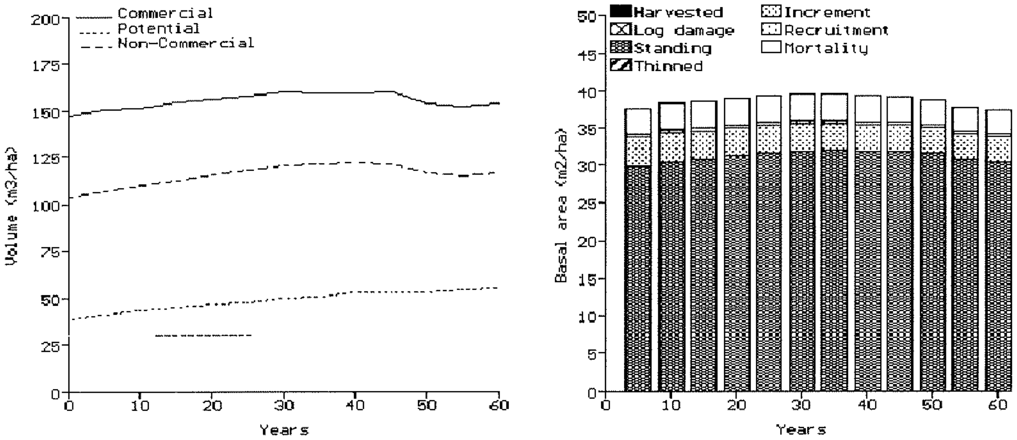


Figure 48 *Simulation of undisturbed forest over a 60-year period by the CAFOGROM model*

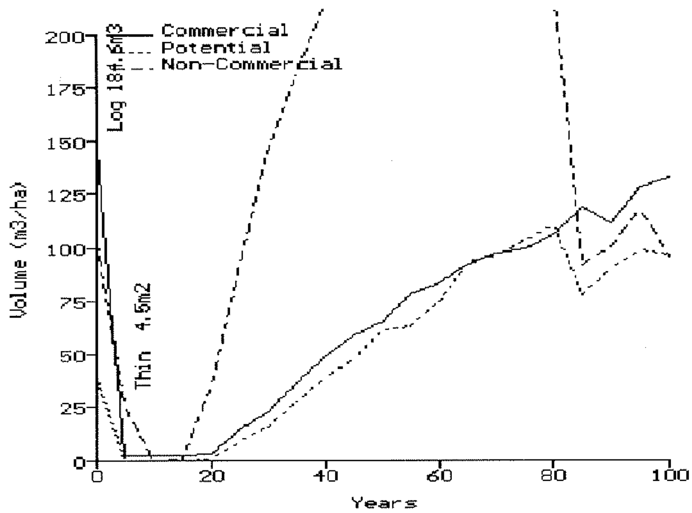


Figure 49 *Ecological succession simulated by CAFOGROM after clearfelling*

model, and comparing it with the expected features of tropical forest dynamics. The following studies illustrate some behavioural tests:

(i) *Dynamic equilibrium of an unlogged stand:* Figure 48 shows how CAFOGROM reproduces the dynamic equilibrium of an undisturbed forest stand. Starting from a database of plots for unlogged forest, and projecting the stand over 60 years, total volumes remain approximately constant. The basal area diagram on the left of the figure shows that this constant state is in fact a dynamic equilibrium in which mortality and increment are closely balanced. This replicates features seen on long-term monitoring plots, such as those at Kade, Ghana (see page 120 and Swaine,

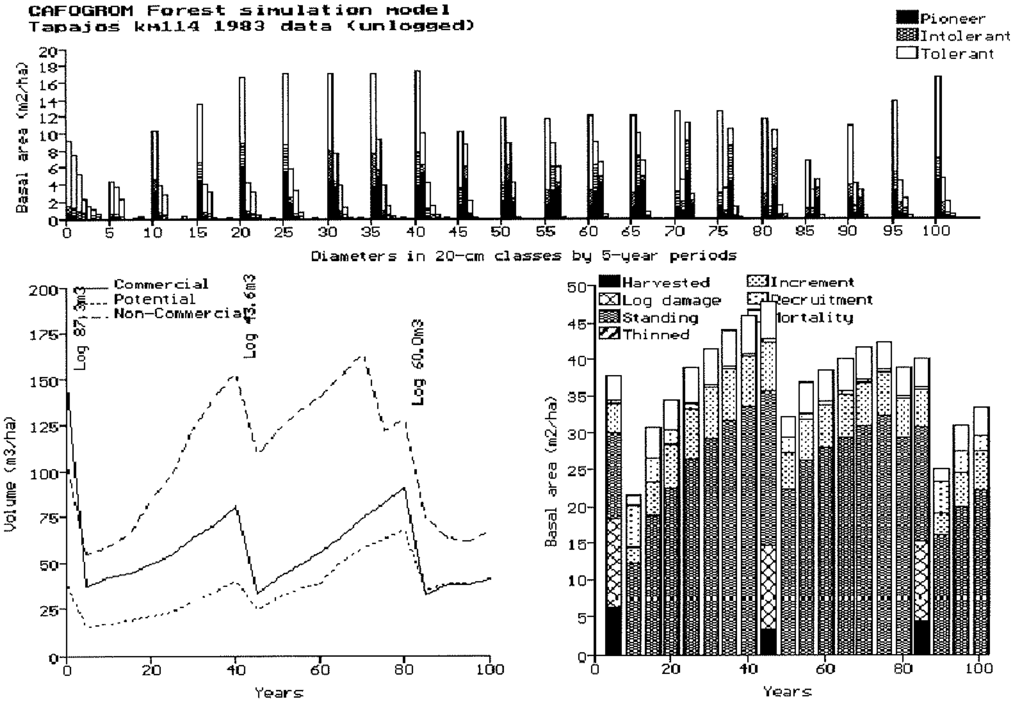


Figure 50 *CAFOGROM simulation of 40-year felling cycle and 60-cm minimum logging diameter, without stand treatment*

1992). In this case, increment and mortality are both at around 3 m^2/ha per 5-year period.

(ii) *Succession and recovery to equilibrium following extreme disturbance:* If the forest is subject to a simulated clear felling, then an interesting picture of ecological succession results (Figure 49). The non-commercial species, mainly comprising fast growing, short lived pioneers, show a rapid recovery, with a population eventually peaking at about 300 m^3/ha (off the top of the graph). The population then crashes, at about 60-70 years of age, as the shorter lived species die out. The slower growing, longer lived species then continue, and begin to show a stabilization of the population after 100 years.

It is notable in this and other examples that typical figures expected for tropical high forest are produced by the simulation. Total volume MAI over 100 years is about 3-4 m^3/ha/yr, with commercial species comprising about 1 m^3/ha/yr. The population recovers from this extreme uniform felling after about 90-100 years. This of course is based only on data from the *terra firme* forest of Tapajos, Brazil. It will be interesting to see how consistently these figures are replicated when cohort models are developed for other forest types in the humid tropics.

(iii) *Logging on a 40 year cycle:* Figure 50 shows the full CAFOGROM screen for a simulation of a typical logging operation. A 40-year felling cycle is applied to previously unlogged forest, with all commercial trees above 60 cm dbh being removed. It can be seen that the initial logging is heavy (about 87 m^3/ha), and subsequent cycles remove somewhat less. The commercial growing stock is reduced to about 35 m^3/ha after logging (bole volume of trees 15 cm dbh plus). A large stock of non-commercial pioneers tends to develop from the initial heavy logging, which however begins to die out in the eightieth year.

(iv) *Logging with volume control and thinning:* Figure 51 shows the same stand, also logged on a 40-year cycle to a minimum girth of 60 cm dbh.

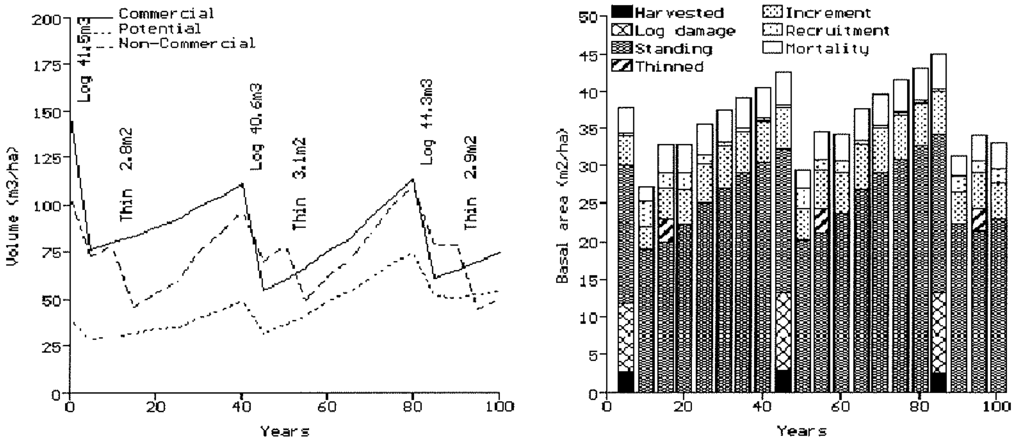


Figure 51 *Stand logged on 40-year cycle with volume control (1 m^3/ha/yr) and post-harvest thinning of non-commercial trees over 40 cm dbh*

However, in this case, logging is constrained to remove only about 40 m^3 (ie. the volume felled equals the felling cycle, or 1 m^3/ha/yr). The stand is thinned after logging to remove non-commercial trees greater than 40 cm dbh. The result is a higher residual growing stock, and less dominance of the stand by non-commercial trees in the post-logging period.

Generally, CAFOGROM shows that mean annual volume increments of slightly in excess of 1 m^3/ha/yr can be sustained provided that the intensity of initial felling of virgin stands is controlled, and moderate thinning is applied to reduce non-commercial stockings. It is an interesting result in that these figures are in no way implicit in the analysis of the data, which is based solely on individual tree diameter increments, mortalities, and recruitment rates. The results agree with most other data on the approximate productivity that may be expected from mixed tropical forest (eg. Dawkins, 1963; UNESCO, 1978:199).

5.2 Growth model application

5.2.1 *Forest stand information as a basis for simulation*

Two situations can be distinguished in the application of forest growth models:

(i) *The model is applied to single stands*: In this situation, the growth model projects the development of either a specified single plot, using the actual data from that plot; or it projects a stand table which has been compiled externally to the program from inventory data or previously summarised results. The GHAFOSIM model (section 3.2.3, page 88) illustrates a model using data compiled from forest inventories into a single aggregate stand table, whose growth is then projected. The PREPSIM program in that example is separate from and external to the simulation process.

Another typical case in this category is the tree position model SYMFOR developed by Young & Muetzelfeldt (1993). This uses measurements including tree coordinates from a specific sample plot as the basis for detailed modelling of a single stand. The data does not need to be summarised in any way, and can be used directly as tree records.

Generally, single stand simulations will be used in research and development applications, where it is necessary to study the behaviour of the model in detail and compare it with growth measurements on specific plots. As GHAFOSIM illustrates, the projection of single stands can also be used a more general planning tool, when the basis is an aggregate stand table representing a large area. However, the validity of the principle of aggregate stand table projection is open to some doubt, and, as has been discussed, should be regarded as acceptable only on a provisional basis.

(ii) *The model is applied to multiple stands*: In this case, forest inventory information, or some kind of permanent compartment register, is used as the basis of the simulation. Forest inventory information is compiled

internally into simulated forest units (SFUs). There will be a correspondence between forest management units such as compartments or coupes and the SFUs on either a one-to-one or one-to-many basis (ie. several SFUs in each compartment).

This type of approach is required for a model that is to be used for detailed management planning of forest resources. Data is not aggregated unduly, and the distinctive features of the growing stock of each management unit can be reflected in differential production schedules and management prescriptions. Vanclay's (1989a) model for North Queensland rainforest falls into this category.

Generally, models used for this type of approach will be either cohort or gap models. A cohort model may typically use a higher level of aggregation than a gap model, with one SFU representing each compartment. In a gap model, there will necessarily be a one-to-one correspondence between inventory plots and SFUs, as the simulation operates at a single tree level for the large tree components. Cohort models may subdivide compartments into several SFUs, but this is probably best done on a flexible basis that may involve some re-merging of cohorts during the simulation, with the sub-SFUs representing different harvesting situations or gap-phases (see page 161 *ff.*).

In the rather early stages of forest management implementation that represents the current situation in most TMF areas, the focus of research and development is on effective and accurate simulation at the level of single stands. However, many countries are now attempting to introduce management inventory in a concerted way, and link it to log control and tracking procedures in a way that naturally suggests the building of a complete forest management information system, as outlined in the next section. It must be expected that over the next decade there will be a considerable movement towards the use of cohort models as part of such systems, and the multiple stand modelling approach will become as normal in TMF applications as it is in the industrial forests of the temperate and boreal zones.

5.2.2 *Growth models and management information systems for TMF*

Figure 52 illustrates the design of a management information system for TMF. It comprises four functional areas, covering forest inventory, growth modelling, management planning, and forest fees and controls on logging. In the real world these functions would evolve independently, and subsequently be integrated into a single system. The various data tables and field names shown in Figure 52 are listed in Table 15.

The role of the growth model, within this FMIS, is to provide estimates of the years when compartments will be felled, and the yields in terms of species group, basal area and numbers removed. The model draws on raw inventory data in the INVTREES file, and uses the relational linkages with the INVPLOTS and COMPARTMENTS databases to organize this data into compartments or SFUs. The applicable

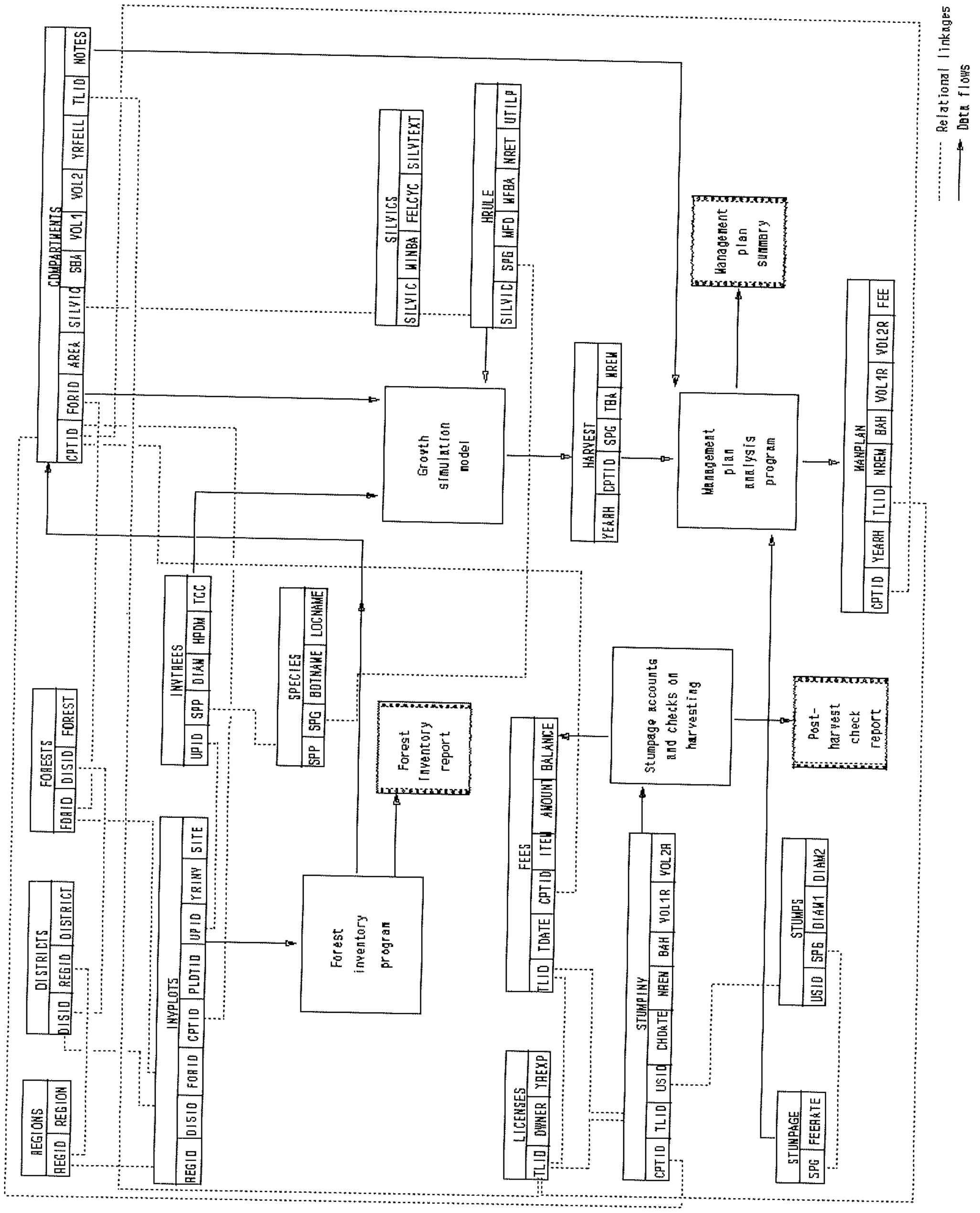


Figure 52 *Data tables and procedures within a typical forest management information system*

Database	Description
REGIONS	A list of names of forest regions (REGION), together with identifer codes (REGID).
DISTRICTS	A list of forest districts. These are sub-units of regions, and the fields define the district and region identity codes (DISID, REGID), together with the district name (DISTRICT).
FORESTS	A list of forest codes and names (FORID, FOREST), with the codes of the districts in which they occur (DISID).
COMPARTMENTS	A list of forest compartments (FORID), with area (AREA), silvicultural system (SILVIC), and a current stock summary including standing basal area (SBA) and estimated commercial volumes above the relavent utilization limits by two categories (VOL1, VOL2). The year the compartment was last felled (YRFELL), the timber licensee identity (TLID), and general management notes (NOTES) are also held.
INVPLOTS	Inventory plot heading records, one per plot. These detail plot location by region, district, forest and compartment, and define a unique internal plot ID used to establish a link to the tree record data (UPID). The year of inventory (YRINV) and general plot data such as site codes (SITE) are held on this record.
INVTREES	These are tree measurement records, one per tree. The UPID field contains the unique plot identity that link tree records for a plot to the plot header record in INVPLOTS. The example shows species code (SPP), diameter (DIAM), height of POM (HPOM) and a tree condition code (TCC) as measured characters.
SPECIES	This is a master table of species codes (SPP), species group codes (SPG), botanical names and local names (BOTNAME, LOCNAME). The species groups in this context are the functional groups recognised by the growth model. A number of different grouping codes could be used for different purposes, such as management and utilization.
SILVICS	This database contains minimum basal area (MINBA) and felling cycle (FELCYC) for a variable number of working circles or different silvicultural systems (SILVIC). The minimum basal area constitutes a basal area below which the stand will be protected even if it has grown beyond the felling cycle. A text description of the working circle is given in SILVTEXT.
HRULE	A set of harvesting rules for each species group and silvicultural system, defined by minimum felling diameter (MFD), number of trees to be retained (NRET), and commercial utilization percent (UTILP), which may be typically 0% or 100% for planning purposes.
HARVEST	Output database from the growth model, detailing cohorts removed with their year of harvest (YEARH), compartment of origin (CPTID), species group (SPG), numbers per km^2 (NREM) and mean tree basal area (TBA).
LICENSES	A list of timber license holders, together with their log mark (TLID), names (OWNER) and the expiry year of the license (YREXP).
FEES	An account ledger for timber fees paid and receivable, showing transaction date (TDATE), relevant compartment number (CPTID), text detailing the transaction (ITEM), the amount received or paid, and the running balance for a given licensee.
STUMPAGE	Fee rates per tree or per m^3 (FEERATE) for each species group (SPG).
STUMPINV	A compartment level record for a post-harvest check inventory of stumps. Gives the check date (CHDATE), and a summary of calculated values for the compartment, including number of stems removed (NREM), basal area harvested (BAH), and volumes removed in two commercial categories (VOL1R, VOL2R). An internal code USID links the STUMPINV records to a number of individual stump records in the STUMPS database.
STUMPS	Individual stump measurements, comprising identification to the species group level (SPG), two diameter measurements at right angles, and a code linking the record to a header record in STUMPINV.

Table 15 *Data tables and fields in a conceptual forest management information system*

harvesting rules are controlled by the SILVICS and HRULE databases. SILVICS specifies a number of silvicultural variants based on felling cycle and minimum stand basal area. For each SILVIC, a list of minimum felling diameters, numbers retained and utilization percentages are given.

The model outputs in the HARVEST routine are collated by a management planning program with other information such as stumpage rates, timber license holders, and the compartment summary database to produce a management plan in both printed and database format.

5.3 Conclusions

This manual has covered the essential steps in developing a predictive model of forest growth and yield for naturally-regenerated tropical mixed forest. These proceed from the preliminary entry, checking, and analysis of permanent sample plot data, through one of two basic routes. Transition matrix, or diameter class projection models can be developed without detailed functional analysis, and using highly aggregated data. The accuracy of such models is doubtful, due to the problem of representing nonlinear dynamic stand-tree interactions within the constraints of an essentially linear model of diameter class transitions. They may nonetheless serve a role in introducing quantitative analysis of yield into high forest management.

A second route is that of developing a functional model incorporating nonlinear dynamic interactions. The cohort representation of the forest stand is simpler in this context than one based on diameter classes. It also lends itself better to deterministic modelling than methods based on individual tree simulation. The structure and some key procedures of such a model have been outlined. The analytical methods and appropriate equations for functions of increment, mortality and recruitment have been considered. The methods of simulating harvesting effects have been described in detail.

A complete model must undergo a process of validation or testing. This is both an ongoing and terminal activity. Ongoing validation is an integral part of the development of a model, whilst terminal validation seeks to define the limits of predictive capability.

To be used effectively, a cohort model needs to be applied to a base of forest inventory information, and requires detailed information about compartment organization and past logging history. Thus such a model both needs and may stimulate the development of a forest management information system.

As Vanclay (1994) has noted, there is no unique or best method of designing a forest growth model. A number of factors influence the choice of basic strategy, and the whole field of research is in too rapid a state of evolution for there to be any certainty that any given method will necessarily be the most effective. The emphasis in this manual has been on building the bridge that is so often lacking, between the field collection of large quantities of raw PSP measurements, and their transformation into an effective and useful synthesis. Its task has been served if it helps the reader to succeed in this objective.

APPENDIX A : CAFOGROM demonstration diskette

A diskette containing a demonstration copy of the CAFOGROM program can be obtained from the author at the following address:

Denis Alder (Consulting) Ltd.
9 Stansfield Close
Headington
OXFORD OX3 8TH

Telephone:	+44 1865 61112
Fax:	+44 1865 60780
Internet:	100574.657@Compuserve.COM

Please photocopy this page and return it, duly completed, to the above address. The request should be made by letter using the letterhead of a forestry research or management institution. As supplies are being made free of charge, *only one diskette will be supplied per institution*. Multiple requests will be referred to the original recipient.

The program requires an MS-DOS environment, Intel 808x6 compatible processor, VGA graphics, and *maths co-processor* in order to run. *It will not run on systems that do not meet these requirements*. The diskette supplied will be in 3½" 1.4 MB format.

Please supply the following information (in block letters):

Full name:	
Position:	
Institution:	
Postal address:	
Fax number:	
E-Mail address:	

Background information on work on growth modelling and permanent sample plots at your institution would be appreciated.

REFERENCES

Adlard, PG (1974) Development of an empirical competition model for individual trees within a stand. Royal College of Forestry, Stockholm. *Research Note* **30**, pp 22-37.

Alder, D (1979) A distance-independent tree model for exotic conifer plantations in East Africa. *Forest Science* **25**(1) 59-71.

Alder, D (1980) Forest volume estimation and yield prediction : Vol. 2 - Yield prediction. *FAO Forestry Paper* **22**/2, 194 pp.

Alder, D (1990) GHAFOSIM : A projection system for natural forest growth and yield in Ghana. Consultancy report to Ministry of Lands & Natural Resources, Ghana. 114 pp.

Alder, D (1991) Data processing for the Budongo Forest inventory. Uganda Forestry Rehabilitation Project internal report. Unpublished. 48 pp.

Alder, D (1992) Simple methods for calculating minimum diameter and sustainable yield in mixed tropical forest. In: *"Wise management of Tropical Forests 1992" Ed. Miller, FR & Adam, KL. Proceedings of the Oxford Conference on Tropical Forests, 30th March-1st April 1992*, Department of Plant Sciences, University of Oxford. pp. 189-200.

Alder, D (1993) Growth and Yield Research in Bobiri Forest Reserve. Forest Research Institute of Ghana, Kumasi. Unpublished Consultancy Report, 71 pp.

Alder, D (1994) The preliminary development of the CAFOGROM forest growth model. ODA/CPATU Silviculture Research Project, Belém, Brazil. Internal consultancy report, 28 pp.

Alder, D; Synnott, TJ (1992) Permanent sample plot techniques for mixed tropical forests. Oxford Forestry Institute, Department of Plant Sciences, University of Oxford, *Tropical Forestry Paper* **25**, 124 pp.

Alemdag, IS (1978) Evaluation of some competition indexes for the prediction of diameter increment in planted White Spruce. Forest Management Institute, Ottawa, Canada. *Information Report FMR-X-108*, 39 pp.

Assmann, E (1970) The principles of forest yield study. Pergamon Press, 506 pp.

Baidoe, JF (1968) An approach to management using sample plots. Commonwealth Forestry Institute, Oxford University. Special subject dissertion, Diploma of Forestry.

Bailey, RL; Dell, TR (1973) Quantifying diameter distributions with the Weibull function. *Forest Science* **19**(2)97-104.

Bella, IE (1970) Simulation of growth, yield and management of aspen. University of British Columbia, Ph. D Thesis. *Dissertation Abstracts* **31**, 6148-B.

Bella, IE (1971) A new competition model for individual trees. *Forest Science* **17**(3)364-372.

Bertalanffy, LV (1957) Quantitative laws in metabolism and growth. *Quarterly Review of Biology* **32**:217-231.

Botkin, DB; Janak, JF; Wallis, JR (1972) Some ecological consequences of a computer model of forest growth. *Journal of Ecology* **60**:849-873.

Brasnett, NV (1953) Planned Management of Forests. George, Allen & Unwin Ltd., London, pp. 128-135.

British Honduras Forestry Department (1949) Departmental Annual Report. Government Printer, Belize City.

Bruner, HD; Moser, JW (1973) A Markov chain approach to the prediction of diameter distribution in uneven-aged forest stands. *Canadian Journal of Forest Research* **3**(3)409-417.

Bryan, MB (1980) Studies of timber growth and mortality in the mixed dipterocarp forests of Sarawak. FAO Project MAL/76/008 Forest Industries Development Project, Field Document. 27 pp.

Buchman, RG (1979) Mortality functions. In: A generalized growth projection system applied to the lake states region. *USDA Forest Service General Technical Report* **NC-49**:47-55.

Buongiorno, J; Hsien Chih Lu (1990) Economic stocking and cutting cycle in a regulated selection forest. *Forest Ecology and Management* **32**:203-216.

Buongiorno, J; Michie, BR (1980) A matrix model of uneven-aged forest management. *Forest Science* **26**(4) 609-625.

Canonizado, JA (1979) Yield tables and site classification in cut-over dipterocarp forests in the Philippines. Paper to Seminar on Management of Dipterocarp Forests, 14 May-5 June 1979, Manila, Philippines. 12 pp + appendices.

Clutter, JL; Fortson JC; Pienaar, LV; Brister, GH; Bailey, RL (1983) Timber Management: A Quantitative Approach. John Wiley & Sons, New York. 333 pp.

Collett, D (1991) Modelling Binary Data. Chapman & Hall, London. 369 pp.

D'Agostino, RB (1971) Linear estimation of the Weibull parameters. *Technometrics* **13**:171-182.

Dawkins, HC (1963) The productivity of tropical high forest trees and their reaction to controllable environment. University of Oxford, *D. Phil.* thesis, 111 pp + appendices.

De Liocourt, F (1898) De l'amenagement des Sapinieres, Bulletin of the Society of Foresters, Franch-Comte Belfort, Besancon, pp. 396-409.

Draper, NR; Smith, H (1966) Applied Regression Analysis. John Wiley & Sons, New York. 407 pp.

Ek, AR; Monserud, RA (1974) FOREST: A computer model for simulating the growth and reproduction of mixed species forest stands. *School of Natural Resources, University of Wisconsin, Research Report* **No. R2635**, 53 pp.

Foli, E (1993) Crown dimensions and diameter growth of some tropical mixed forest trees in Ghana. University of Aberdeen, M.Phil thesis, 173 pp.

Gourlet-Fleury, S (1992) Indices de competition: Les possibilites d'application a la gestion en foret dense tropicale humide. Laboratoire de Biometrie, Universite Claude Bernard, Lyon, Publication No. URA CNRS 243, 30 pp + refs, apps.

Green, PE (1976) Mathematical Tools for Applied Multivariate Analysis. Academic Press, New York. 376 pp.

Hawthorne, WD (1993) Forest regneration after logging. Overseas Development Administration, London. *ODA Forestry Series* **3**, 52 pp.

Husch, B; Miller, CI; Beers, TW (1982) Forest Mensuration, Third Edition. John Wiley & Sons, New York. 402 pp.

Johnson, MS; Chaffey, DR (1973) An inventory of the Chiquibul Forest Reserve, Belize. Land Resources Division, Overseas Development Administration. Land Resources Study No. 14, 87 pp.

Jonkers, WBJ (1987) Vegetation structure, logging damage and silviculture in a tropical rainforest in Surinam. Wageningen Agricultural University, Ph.D. thesis, 172 pp.

Kofod, EO (1982) Stand table projections for the mixed dipterocarp forests of Sarawak. Forest Department, Kuching, Sarawak, Malaysia. *[FAO Report] FAO/MAL/76/008 Field Document* **No. 9**, 107 pp.

Korsgaard, S (1992) An analysis of growth parameters and timber yield prediction. The Council for Development Research, Copenhagen. Consultancy Report, 120 pp + appendices.

Korsgaard, S (1993) A user's guide to the computer programmes used in processing and tabulating forest research data from the CPATU permanent sample plots. *CPATU/EMBRAPA/IICA Technical Paper* **No. 1**, 100 pp.

Korsgaard, S (1994) A manual for the standtable projection simulation model. International Tropical Timber Organization/Forest Department, Kuching, Sarawak. Project Field Document. 66 pp.

Leak, WB (1965) The J-shaped probability distribution. *Forest Science* **11**(4)405-409.

Leak, WB (1970) Sapling stand development: A compound exponential process. *Forest Science* **16**(2)177-180.

Leslie, PH (1945) On the use of matrices in certain population mathematics. *Biometrika* **33**:183-212.

Leslie, PH (1948) Some further notes on the use of matrices in population mathematics. *Biometrika* **35**:213-245.

Loetsch, F; Zohrer, F; Haller, KE (1973) Forest Inventory, Volume 2. BLV Verlagsgesellschaft, Munich. 469 pp.

Maginnis, S (1994) Understorey scoring during pre-exploitation stock survey. *Commonwealth Forestry Review*, in press.

Makela, A; Hari, P (1986) Stand growth model based on carbon uptake and allocation in individual trees. *Ecological modelling* **33**:205-229.

Mengin-Lecreulx, P (1990) Simulation de la croissance d'un peuplement de foret dense: Le cas de la foret de Yapo (Cote d'Ivoire). Internal research report, CTFT, Paris. 55 pp.

Mervart, J. (1972) Growth and mortality rates in the natural high forest of western Nigeria. *Nigeria Forestry Information Bulletin (New Series)* **no. 22**.

Meyer, HA (1933) Eine mathematische-statistische Untersuchung über den Aufbau des Plenterwaldes. *Schweizer Zeitschrift fur Forstwirtschaft* **84**:33-40, 88-103, 124-131.

Meyer, HA (1944) A correction for systematic error occurring in the application of the logarithmic volume equation. *Pennsylvania State University Forest Research Paper* **7**.

Meyer, HA (1952) Structure, growth, and drain in balanced uneven-aged forests. *Journal of Forestry* **50**:85-92.

Michie, BR; Buongiorno, J (1984) Estimation of a matrix model of forest growth from re-measured permanent plots. *Forest Ecology and Management* **8**:127-135.

Monserud, RA; Ek, AR (1974) Plot-edge bias in forest stand growth simulation models. *Canadian Journal of Forest Research* **4**(4)419-423.

Nelson, W (1982) Applied Life Data Analysis. John Wiley & Sons, New York. 634 pp.

Nicholson, DI (1958) Analysis of logging damage in tropical rain forest, North Borneo. *Malay Forester* **21**:235-245.

Nicholson, DI (1979) The effects of logging and treatment on the mixed dipterocarp forests of south-east Asia. FAO, Rome, Publication FO:MISC/79/8. 65 pp.

Oliver, CD; Larson, BC (1990) Forest Stand Dynamics. McGraw-Hill Inc., New York. 467 pp.

Opie, JE (1968) Predictability of individual tree growth using various definitions of competing basal area. *Forest Science* **14**(3)314-323.

Osafo, ED (1970) The development of silvicultural techniques applied to natural forests in Ghana. Forest Products Reserach Institute, Kumasi, Technical Note 13, 15 pp + appendices.

Owen, RA (1987) A preliminary report for Tinte Bepo Forest Reserve with recommendations for management. Forest Inventory Project, Forestry Department, Kumasi, Internal Document. 27 pp + appendices.

Philip, MS (1994) Measuring Trees and Forests, Second Edition. CAB International, Wallingford, UK. 336 pp.

Poore, D; Burgess, PF; Palmer JR; Rietbergen S; Synnott, TJ (1989) No timber without trees. Earthscan Publications Ltd., London. 252 pp.

Press, WH; Flannery, BP; Teukolsky, SA; Vetterling, WT (1989) Numerical Recipes: The art of scientific computing (FORTRAN version). Cambridge University Press, 702 pp.

Rai, SN (1979) Rate of diameter growth and age-diameter relationship of *Vitex altissima* and *Lannea coromandelica* in moist deciduous forests of Karnataka. *Indian Journal of Ecology* **6**(1)20-29.

Schmitt, L; Bariteau, M (1989) Gestion de l'ecosysteme forestier Guyanais: etude de la croissance et de la regeneration naturalle (Dispositif de Paracou). *Bois et Forets des Tropiques* **220**:3-23.

Seber, GAF (1977) Linear Regression Analysis. John Wiley & Sons, New York. 465 pp.

Shannon, RE (1975) Systems Simulation: The art and science. Prentice-Hall Inc., Englewood Cliffs, New Jersey. 387 pp.

Sievanen, R; Burk, TE; Ek, AR (1988) Construction of a stand growth model utilizing photosynthesis and respiration relationships in individual trees. *Canadian Journal of Forest Research* **18**:1027-1035.

Silva, JNM (1989) The behaviour of the tropical rainforest of the Brazilian Amazon after logging. D.Phil. thesis, Oxford University. 302 pp.

Silva, JNM; de Carvalho, JOP; Lopes, JCA; Almeida BF; Costa DHM; de Oliveira, LC; Vanclay, JK; Skovsgaard, JP (1994) Growth and yield of a tropical rainforest in the Brazilian Amazon 13 years after logging. *Forest Ecology and Management, in press.*

Singh, KD (1981) A review of FAO's contribution to the study of growth and yield in mixed tropical forests. Paper to IUFRO S4.01 meeting, Los Banos, Phillipines. 9 pp.

Swaine, MD (1992) Population dynamics of moist tropical forest at Kade, Ghana. In: *Actes de l'atelier sur l'amenagement de l'ecosysteme forestier tropical humide, Cayenne, March 1990. [Eds. Maitre, HF; Puig, H].* MAB/UNESCO/IUFRO/FAO. pp. 40-61.

Swaine, MD; Hall, JB; Alexander, IJ (1987) Tree population dynamics at Kade, Ghana (1968-1982). *Journal of Tropical Ecology* **3**:331-345.

Tang, HT (1976) Studies on the stocking and growth of some mixed dipterocarp forests after logging in peninsular Malaysia. Jesus College, Oxford University. M.Sc. thesis. 172 pp.

UNESCO (1978) Tropical forest ecosystems: A state of knowledge report. UNESCO, Paris, *Natural resources research* **XIV,** 683 pp.

Urban, DL; Bonan, GB; Smith, TM; Shugart, HH (1991) Spatial applications of gap models. *Forest Ecology and Management* **42**:95-110.

Usher, MB (1966) A matrix approach to the management of renewable resources, with special reference to selection forests. *Journal of Applied Ecology* **3**(2)355-367.

Usher, MB (1969) A matrix approch to the management of renewable resources with special reference to selection forests - two extensions. *Journal of Applied Ecology* **6**(2)347-348.

Valentine, HT (1988) A carbon-balance model of stand growth: A derivation employing pipe-model theory and the self-thinning rule. *Annals of Botany* **62**:389-396.

Valentine, HT (1989) A carbon-balance model of tree growth with a pipe-model framework. In *"Process modelling of forest growth responses to environmental stress" Dixon, RK; Meldahl, RS; Ruark, GA; Warren, WG [Eds.],* Timber Press, Portland, Oregon. pp 33-40.

Vanclay, JK (1989a) A growth model for North Queensland rainforests. *Forest Ecology and Management* **27**:245-271

Vanclay, JK (1989b) Modelling selection harvesting in tropical rain forests. *Journal of Tropical Forest Science* **1**(3)280-294.

Vanclay, JK (1991) Mortality functions for north Queensland rainforests. *Journal of Tropical Forest Science* **4**:15-36.

Vanclay, JK (1994) Modelling Forest Growth and Yield: Applications to Mixed Tropical Forests. CAB International, Wallingford, UK. 280 pp.

Vanclay, JK; Henry, NB (1988) Assessing site productivity of indigenous cypress pine forest in southern Queensland. *Commonwealth Forestry Review* **67**(1)53-64.

Waring, RH; Schroeder, PE; Oren, R (1982) Application of the pipe model theory to predict canopy leaf area. *Candian Journal of Forest Research* **12**:556-560.

West, PW (1980) Use of diameter and basal area increment in tree growth studies. *Candian Journal of Forest Research* **10**:71-77.

Wilkinson, L (1988) SYSTAT: The system for statistics. Systat, Inc., Evanston, Illinois.

World Bank (1992) Federal Republic of Nigeria - Forest Sector Review. Confidential Report No. 10744-UNI, 87 pp + maps.

Wykoff, WR (1990) A basal area increment model for individual conifers in the northern Rocky Mountains. *Forest Science* **36**:1077-1104.

Young, A; Muetzelfeldt, R (1993) SYMFOR 1.0 : An individual-based demonstration model of tropical forest dynamics. Institute of Ecology and Resource Management, Edinburgh University Internal Report. 28 pp.

Zeide, B (1990) Structure of growth equations. *University of California Division of Agriculture and Natural Resources, Bulletin* **1927**:349-354.

Zohrer, F (1970) The Beta-distribution for best fit of stem diameter distributions. Paper to 3rd IUFRO Conference of the Advisory Group of Forest Statisticians, Jouy-en-Josas, 7-11 Sept. 1970. Institut National de la Recherche Agronomique, Paris. *INRA Publication* **72-3**:91-106.

APPENDIX B : Listing of the CAFOGROM program

This listing is of a program written in C for the Borland Turbo C 2.0 compiler, and assumes the availability of library software from that environment.

```
/* CAFOGROM : CPATU Amazon Forest Growth Model

This model has been written by Dr. Denis Alder for CPATU under the
ODA Silviculture Research Project.  It uses data from Tapajos Forest
Research plots at km67 and km114, and is designed as a teaching and
demonstration model for forest management and growth modelling methods.
*/

/* library files */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <graphics.h>
#include <conio.h>
#include <string.h>

/* Protypes of procedures/functions requiring arguments */
void axis(int,   int, int, float, float, char *);
int nTtic(int);
int nGet( int, int, int, int *, int);
int getdata(void);
int l2b(const void *, const void *);
int b2l(const void *, const void *);
float voleqn(float);
int smallest(int, int, int);

/* array limits  definitions */
#define NMOD 16
#define GUILDS 3
#define COMAX 2000
#define PRGMAX 300
/* graphics screen limits */
#define XMIN 0
#define XMAX 639
#define YMIN 479
#define YMAX 0
#define XSPLIT 320
#define YSPLIT 180
#define DDMAX 20
#define BAMAX 50
#define VOLMAX 200
#define RUNLIMIT 5
/* other model constants */
#define TSTEP 5
const char guilds[3][3]={"PI","LD","SB"};
/* global arrays and data structures */
float tbf[NMOD][4];
float psr[NMOD][4];
float mit[NMOD];
float maxtba[NMOD];
struct recruit { int guild; int gp; int model; float p; } ;
struct recruit prg[PRGMAX];
struct cohort {int gp; int model; int dc; float tba; int cp; float nha;};
struct cohort *sfu, *sfu0;
int nco, nco0, nprg, logging=0;
```

```
char modlets[NMOD];
/* global variables used for graphics display */
int gdrv, gmode;
int ux0,ux1,uy0,uy1, uyb;
int vx0,vx1,vy0,vy1;
int wx0,wx1,wy0,wy1;
int gdrv=DETECT, gmode;
int ThinFlag=0, LogFlag=0;
/* basal area at start of each step, mortality and recruitment ba, cut or
     thinned ba, and ba at end of time step */
float sba0, sbai, mba, rba, cba, sba5, hBA, hVol, tVol[3], AreaLost,thBA,
          lostBA ;
/* time and felling cycle variables */
int Tcycle, Tsim, Tmax, FelCyc, ThinYr, nData, MinD, Nseed,
     ThinD, ThinGp, LogOpt, FirstLog, VolCon, VolRem, ThinFq, ThinBA;
int YeaL, YeaT;
/* main program */
main()
{  char key;
   InitScreen();                 /* initial CAFOGROM screen */
   Setup();                      /* read coefficient arrays */
   key=getch();                  /* wait for any key */
   /* Display run option screen and execute simulation. */
   while(RunOpt())
        if(getdata()) simulate();
   clrscr();
}

int RunOpt()
/* Displays a screen of options for the simulation run.  Returns 1 when
     options are set.  Returns zero if ctrl-Q is pressed. */
{
   /* field definitions and default values */
   int Fdef[3][14]={{50, 45, 30, 30, 30, 66, 66, 66, 45, 30, 30, 66, 66, 30},
                    { 3,  9, 10, 11, 12, 10, 11, 12, 15, 16, 17, 16, 17, 19},
                    { 1,  1,  3,  3,  3,  3,  1,  3,  1,  2,  2,  2,  2,  3}};
   static int nf=14,
          Fval[14]={ 1,  1, 30, 45,100, 30,  2,100,  0,  0,  0,  0,  0, 40};
   static int nrun=0;
   int i,j,k;
   /* display text */
   restorecrtmode();
   clrscr();
   /* exit program after a few runs, otherwise DOS gets screwed up */
   if(nrun++>=RUNLIMIT){printf("CAFOGROM finished OK \n \n"); exit(0);};
   printf("CAFOGROM : CPATU Amazon Forest Growth Model  -  Simulation Options");
   gotoxy(1,3);
   printf("Select forest used as a basis for simulation\n");
   printf("1  Tapajos km67 plots, 1981 measurement (logged 1979)\n");
   printf("2  Tapajos km114 control plots, 1983 measurement (unlogged forest)\n");
   printf("3  Tapajos km114 treated plots, 1983 measurement (logged 1982)\n");
   printf("\n");
   printf("Harvesting options:\n");
   printf("   Log (0) None (1) C (2) C+P species\n");
   printf("   Felling cycle (years)          First logging year\n");
   printf("   Minimum diameter (cm)             Volume in (1) m3 (2) %\n");
   printf("   Seed trees left (N/km2)         Volume to remove \n");
   printf("\n");
   printf("Thinning options:\n");
   printf("   Thin (0) None (1) NC (2) NC+P trees\n");
   printf("   Year of first thinning          Residual BA (m2/ha)\n");
   printf("   Thinning frequency, years          Minimum diameter (cm)\n");
```

```
    printf("\n");
    printf("Run time (years)\n");
    printf("\n \n");
    printf("Arrow keys move between fields.  <Backspace> clears a field.\n");
    printf("<Enter> executes simulation with current options.\n");
    printf("<Esc> suspends or quits simulation while running.\n");
    printf("<ctrl-Q> returns to DOS");
    /* display current field values */
    for (i=0; i<nf ;i++)
        {
        nGet(Fdef[0][i], Fdef[1][i], Fdef[2][i], &Fval[i], 0);
        };
    /* loop to input field values. nGet returns :
    (1) up arrow, (2) down arrow, (3) <enter>, (4) <ctrl-Q> */
    i=0; k=0;
    while(k < 3)
        {
        k=nGet(Fdef[0][i], Fdef[1][i], Fdef[2][i], &Fval[i], 1);
        switch(k)
            {
            case 1:   /* up arrow */
                        if(i>0) i--; else i=nf-1;
                        break;
            case 2:   /* down arrow */
                        if(i<nf-1) i++; else i=0;
                        break;
            case 3:   /* <enter> - run simulation */
                        break;
            case 4:   /* <ctrl-Q> - return to main program with value zero */
                        return 0;
            };
        };
    /* put field values in appropriate variables */
    nData=Fval[0];
    LogOpt=Fval[1];
    FelCyc=Fval[2];
    MinD=Fval[3];
    Nseed=Fval[4];
    FirstLog=Fval[5];
    VolCon=Fval[6];
    VolRem=Fval[7];
    ThinGp=Fval[8];
    ThinYr=Fval[9];
    ThinFq=Fval[10];
    ThinBA=Fval[11];
    ThinD=Fval[12];
    Tmax=Fval[13];
    /* reset global run timers, flags, accumulators */
    ThinFlag=0; LogFlag=0; YeaL=0; YeaT=0;
    gotoxy(1,24);
    return 1;
}
```

```
int nGet( int Col, int Row, int Flen, int *Fval, int mode)
/* retrieves a positive integer value from a field of Flen characters
    located at (Col,Row) into Fval (which contains a default value).
    if 'mode' is 0, only the current value in Fval is displayed.  If it is 1,
    then the routine waits for input.  The value returned is (1) or (2) if
    the field is terminated by up or down arrows, (3) if <enter>, (4) if
    <ctrl-Q>.  Input of characters into a full field, or pressing other keys,
    produces a tone. */
{
    int i,k,u,v, more, Key ;
    char a[10],b;
    const char Up=28, Down=29, Enter=13, ctrlQ=17, Cursor=220;
    u=*Fval;
    itoa(u,a,10);
    /* display brackets [  ] for the field */
    gotoxy(Col-1,Row);
    putch('[');
    gotoxy(Col+Flen,Row);
    putch(']');
    /* display current value */
    gotoxy(Col,Row);
    cputs(a);
    /* if mode is 0, return with value 2 (next field) at this point */
    if(mode= =0) return 2;
    /* loop to process input characters */
    k=0;
    more=1;
    while (more)
        {
        /* output current string */
        gotoxy(Col,Row);
        cputs(a);
        /* get character and examine it */
        gotoxy(Col+k,Row);
        b=getch();
        if (b> =48 && b< =57)
            {
            a[k]=b;
            if(k<Flen-1) k++; else beep();
            }
        else
            {
            /* check for permitted control characters */
            switch (b)
                {
                case '\x0':
                    /* up or down arrow keys */
                    switch (getch())
                        {
                        case 'H':  more=0; Key=1; break;
                        case 'P':  more=0; Key=2; break;
                        default:          beep();
                        };
                    break;

                case '\r':       more=0; Key=3; break;
                case '\x11':     more=0; Key=4; break;
                case '\x08':     /* delete field */
                                     gotoxy(Col,Row);
                                     for(i=0; i<Flen; i++) putch(' ');
                                     k=0;
                                     memset(a,'\0',10);
```

```
                                        break;
                    default:            beep();
                    };
                };
            };
        /* replace input value and return */
        a[Flen]='\0';
        *Fval=atoi(a);
        return Key;
};

beep()
/* sounds a beep */
{
        sound(800);
        delay(50);
        nosound();
};

int getdata()
/* opens and reads SFU data file.  nData is set in the RunOpt routine */
{
        int i,j,gp,m,dc,cp;
        char *ipf[]={"1202-81.CFD","1401-83.CFD","1121-83.CFD"};
        int LogHist[]={1,0,1};
        char modc;
        float tba,nha;
        FILE *dfl;
        /* open data file */
        dfl=fopen(ipf[nData-1],"rt");
        logging=LogHist[nData-1];
        /* allocate space for cohort array and working copy */
        sfu=malloc(sizeof(struct cohort)*COMAX);
        sfu0=malloc(sizeof(struct cohort)*COMAX);
        /* read data */
        for( i=0; ! feof(dfl); i++)
            {
            /* read group, model, d-class, TBA, crown class, NHA */
            fscanf(dfl,"\n%d %c %d %f %d %f",&gp, &modc, &dc, &tba, &cp, &nha);
            /* get index number for model */
            for(m=0; modc!=modlets[m] && m<NMOD ;m++);
            if(m>=NMOD)
                {
                printf("\n\nError in data at line %d in %s !", i+1, ipf[nData]);
                exit(1);
                };
            /* add details to cohort list */
            sfu0[i].gp=gp;
            sfu0[i].model=m;
            sfu0[i].dc=dc;
            sfu0[i].tba=tba;
            sfu0[i].cp=cp;
            sfu0[i].nha=nha;
            /* check cohort array space */
            if(i>=COMAX)
                {
                printf("\n\nToo many cohorts to run simulation (Increase COMAX) !");
                exit(1);
                };
            };
        nco0=i-1;
        fclose(dfl);
        return 1;
    }

    simulate()
    /* growth modelling routine */
    {
        /* make working copy of cohort array */
        memcpy(sfu, sfu0, sizeof(struct cohort)*COMAX);
        nco=nco0;
        /* set up display frames and initialize graphic variables */
        GraphFrame();
        Tsim=0;
        GraphOut();
        /* simulation loop */
        for(Tsim=TSTEP; Tsim< =Tmax && !EscKey(); Tsim+ =TSTEP)
            {
            /* The following statements simulate, in turn, growth, mortality,
            and recruitment.  The time within the currenct cycle (Tcycle) is
            checked to see if harvesting or thinning is due.  Crown classes
            are adjusted for liberation or suppression.  Similar cohorts are
            merged and empty ones removed.  The status of the forest is output
            to the display.  The loop continues until Tmax or the esc key is
            pressed */
    /*      Audit();    used only for diagnostic purposes */
            Growth();
            Mortality();
            Recruitment();
            if(LogCheck()) Harvest(); else {logging=0; hVol=0.0; hBA=0.0;};
            if(ThinCheck()) Thinning(); else thBA=0.0;
            CrownAdjust();
            MergeCohorts();
            GraphOut();
            YeaT+ =TSTEP;
            YeaL+ =TSTEP;
            };
        /* wait for a keystroke, then save display and revert to
        command mode */
        gotoxy(1,1);
        getch();
    }

    int LogCheck()
    /* checks to see if a logging is due */
    {
        /* logging option set to zero - no logging */
        if(LogOpt= =0)
            LogFlag=0;
        else
            /* see if years elapsed corresponds to felling cycle */
            if(YeaL> =FelCyc | | Tsim= =FirstLog)
                {
                /* reset logging and thinning cycle timers */
                YeaL=0;
                YeaT=0;
                LogFlag=1;
                }
            else
                LogFlag=0;
        return LogFlag;
    }
```

```
int ThinCheck()
/* checks to see if a thinning is due */
{
    /* thinning option set to zero - no thinning */
    if(ThinGp==0)
        {
        ThinFlag=0;
        return 0;
        };
    /* test for first thinning */
    if(YeaL==ThinYr)
        {
        YeaT=0;
        ThinFlag=1;
        return 1;
        }
    /* test for thinning cycle */
    if(ThinFq==0)
        {
        ThinFlag=0;
        return 0;
        };
    if(YeaT==ThinFq && YeaL>ThinYr)
        {
        ThinFlag=1;
        YeaT=0;
        return 1;
        };
    /* no thinning required under any condition tested */
    ThinFlag=0;
    return 0;
}

Audit()
/* checks the SFU array for internal consistency and reports errors */
{
    int i, erflag;
    struct cohort *co;
    erflag=0;
    for (i=0; i<=nco ; i++)
        {
        co=&sfu[i];
        if(co->gp<1      || co->gp>9       ){erflag=1; break;};
        if(co->model<0   || co->model>15   ){erflag=2; break;};
        if(co->tba<19    || co->tba>40000){erflag=3; break;};
        if(co->cp<0      || co->cp>1       ){erflag=4; break;};
        if(co->nha<=0    || co->nha>1000  ){erflag=5; break;};
        };
    if(erflag!=0)
        {
        gotoxy(1,1);
        printf("AUDIT error %d at SFU= %d",erflag,i);
        };
}
```

```
Growth()
/* Applies the tree basal area increment function to each cohort, using
    coefficient values for each species model.  This routine also gets
    stand basal area at the beginning of the period, sba0, and increment
    ba, sbai */
{
    int i, m; float tbai; float tba,*b;
    sba0=0; sbai=0;
    for(i=0; i<=nco ; i++)
        {
        m=sfu[i].model;
        tba=sfu[i].tba;
        sba0+=tba*sfu[i].nha;
        b=&tbf[m];
        tbai=TSTEP*exp(b[0] + b[1]*sfu[i].cp + b[2]*log(tba) + b[3]*tba);
        sfu[i].tba+=tbai;
        sbai+=tbai*sfu[i].nha;
        };
    sba0*=0.0001;
    sbai*=0.0001;
}

Mortality()
/* applies the mortality function to each cohort, and calculates total
    mortality basal area.  The 'logging' variable may be 0 or 1, depending
    on whether stand was logged in last 5-year period */
{
    int i, j, m; float tbai, nha;
    mba=0;
    for(i=0; i<=nco ; i++)
        {
        m=sfu[i].model;
        if (sfu[i].tba>maxtba[m])
            sfu[i].nha=0.0;
        else
            {
            j=sfu[i].cp+2*logging;
            nha=sfu[i].nha;
            mba+=(nha-nha*psr[m][j])*sfu[i].tba;
            sfu[i].nha*=psr[m][j];
            };
        };
    mba*=0.0001;
}

Recruitment()
/* adds recruit cohorts to the SFU array.  The total number of recruits
    depends on basal area at start of the period (sba0).  This relative
    allocation to pioneers, light demanders, and shade bearers is also
    a function of number of recruits and hence BA */
{
    int i,j,k,m;
    float nin, p[3], f;
    if(sba0<28)   nin=1526-52.6*sba0; else nin=50;
    p[0]=1/(1+1/(0.0431*pow(nin,0.456)));
    p[1]=1/(1+1/(0.0529*pow(nin,0.309)));
    p[2]=1-(p[1]+p[2]);
    rba=0;
    for (i=nco+1, j=0; j<nprg; i++, j++)
    {
        sfu[i].gp=prg[j].gp;
        m=prg[j].model;
```

```
                sfu[i].model=m;
                f=mit[m];
                sfu[i].dc=0;
                sfu[i].tba=mit[m];
                sfu[i].cp=0;
                k=prg[j].guild;
                sfu[i].nha=nin*p[k]*prg[j].p;
                rba+=sfu[i].tba*sfu[i].nha;
                if(i>=COMAX)
                  {
                  gotoxy(1,1);
                  printf("Array space limit reached for recruits - Increase COMAX !");
                  getch();
                  return;
                  };
        };
        nco=i-1;
        rba*=0.0001;
}

Harvest()
/* The harvesting routine is called each felling cycle and works as
        follows:
        (1)  An index of commercial cohorts is created.
        (1)  Cohorts are sorted from largest to smallest.
        (2)  The index is skipped until Nseed seed trees have been left.
        (3)  Remaining commercial trees have Nha set to zero, and volume
                added to hVol.
*/

{
        int i, j, k, n, nc;
        float stree, BAmin, tvol, pct, f;
        int *ix, hgp;
        struct cohort *co;
        ix = (int *)malloc(nco*sizeof(int));
        BAmin=(float)MinD*MinD*0.7854;
        /* make index of commercial trees and calculate total stand volume */
        k=-1;
        for (j=0; j<3; j++)tVol[j]=0.0;
        hgp=( LogOpt==1 ? 2 : 4);
        for(i=0; i<=nco; i++)
              {
              co=&sfu[i];
              if(co->gp<=hgp && co->tba>BAmin)
                    {
                    k++;
                    ix[k]=i;
                    };
              j= (co->gp > 4 ? 2 : (co->gp > 2 ? 1 : 0));
              tVol[j]+=voleqn(co->tba)*co->nha;
              };
        tvol=tVol[0]+tVol[1]+tVol[2];
        nc=k;
        if(nc<0) return ;  /* no trees eligible for harvest */
        qsort(ix, nc, sizeof(int), b2l);
        /* skip over seed trees */
        stree=0.0;
        for (k=1; k<=nc && stree<Nseed; k++)stree+=sfu[ix[k]].nha*100;
        if(k>nc) return ;  /* no trees eligible for harvest */
        /* harvest eligible trees */
        hVol=0;
```

```
    hBA=0;
    pct=100.0/tvol;
    for (i=k; i<=nc; i++)
        {
        co=&sfu[ix[i]];
        hVol+=voleqn(co->tba)*co->nha;
        if(VolCon==1)
            if(hVol>=VolRem)break;
        else
            if(hVol*pct>=VolRem)break;
        hBA+=co->tba*co->nha;
        co->nha=0;
        };
    /* adjustment of stock for skid trails, landings etc. */
    hBA*=0.0001;
    sba0-=hBA;
    AreaLost= pow((hVol/tvol),0.575);
    lostBA=0;
    for(i=0 ; i<=nco ; i++)
        {
        co=&sfu[i];
        lostBA+= co->tba*co->nha*AreaLost;
        sfu[i].nha*=(1.0-AreaLost);
        }
    lostBA*=0.0001;
    logging=1;
    LogFlag=1;
    free(ix);
}

Thinning()
/* Thinning is simulated by removing trees of Non-Commercial or Potential
    groups down to a specified basal area and diameter. */
{
    int i, k, nc, minba, tgp, *ix;
    struct cohort *co;
    float BA, BAleft,BAmin;
    ix = (int *)malloc(nco*sizeof(int));
    tgp= (ThinGp==1 ? 5 : 3);
    /* index and sort trees eligible for thinning from largest to smallest */
    BAmin=(float)ThinD*ThinD*0.7854;
    k=-1;
    for(i=0; i<=nco; i++)
        {
        co=&sfu[i];
        if(co->gp>=tgp && co->tba>BAmin)
            {
            k++;
            ix[k]=i;
            };
        };
    nc=k;
    if(nc<0) return ;  /* no trees eligible for harvest */
    qsort(ix, nc, sizeof(int), b2l);

    /* perform thinning by setting stocking to zero.  exit when required
        post-thinning BA achieved or list of eligible trees exhausted */
    BAleft=sba0;
    thBA=0.0;
    for(i=0; i<=nc && BAleft>ThinBA ; i++)
        {
        co=&sfu[ix[i]];
```

```
            BA=co->tba*co->nha*0.0001;
            BAleft-=BA;
            thBA+=BA;
            co->nha=0.0;
            };
        sba0=BAleft;
        ThinFlag=1;
}

CrownAdjust()
/* Liberates or suppresses trees depending on the basal area of canopy
        trees  */
{
        int i, j, k, n, nc;
        float limba, cba;
        int *ix;
        ix = (int *)malloc(nco*sizeof(int));
        /* calculate basal area of canopy trees */
        cba=0;
        for(i=0; i<=nco; i++)
            {
            if(sfu[i].cp==1)
                {
                cba+=sfu[i].tba*sfu[i].nha;
                };
            };
        cba*=0.0001;
        /* get limiting basal area for canopy trees */
        limba=sba0*(0.80-0.0072*sba0);
        if(cba>limba)
            {
            /* too many canopy trees - create an index of all CP=1 cohorts and
                sort it from smallest to largest */
            CPindex(ix,&nc,1);
            qsort((void *)ix, (size_t)nc, (size_t)sizeof(int), l2b);
             /* downgrade cohorts and decrement cba until limba passed */
            for(j=0; j<=nc && cba>limba; j++)
                {
                k=ix[j];
                sfu[k].cp=0;
                cba-=sfu[k].tba*sfu[k].nha*0.0001;
                };
            }
        else
            {
            /* too few canopy trees - create an index of all CP=0 cohorts and
                sort from largest to smallest */
            CPindex(ix,&nc,0);
            qsort(ix, nc, sizeof(int), b2l);
             /* upgrade trees and increment cba until limba passed */
            for(j=0; j<=nc && cba<limba; j++)
                {
                k=ix[j];
                sfu[k].cp=1;
                cba+=sfu[k].tba*sfu[k].nha*0.0001;
                };
            };
        free(ix);
/* end of routine CrownAdjust */
}
```

```
CPindex(int *ix, int *nc, int cp)
/* creates an index of cohorts with crown position code 'cp'.  'nc' returns
    the number of cohorts in the index */
{
    int i, k;
    k=-1;
    for(i=0; i<=nco; i++)
        {
        if(sfu[i].cp==cp)
            {
            k++;
            ix[k]=i;
            };
        };
    *nc=k;
}

int l2b(const void * ix1, const void * ix2)
/* used by 'qsort' library routine to sort index ix from smallest to biggest */
{
    return sfu[*(int *)ix1].tba-sfu[*(int *)ix2].tba;
}

int b2l(const void * ix1, const void * ix2)
/* used by 'qsort' library routine to sort index ix from smallest to biggest */
{
    return sfu[*(int *)ix2].tba-sfu[*(int *)ix1].tba;
}

MergeCohorts()
/* This routine assigns a 2-cm diameter class to each cohort and then
    merges cohorts with same group, model, d-class and crown class.  It also
    deletes cohorts where the N/ha is less than 0.01.  The objective is to
    avoid an unnecessary proliferation of cohorts.  During merging, the mean TBA
    of the merged cohort is the weighted average of the two constituents.  The
    N/ha of the merged cohort is the sum of the constituents */
{
    int i, j, k, L, nco0; struct cohort *a, *b, *c;
    long ax,bx;
    float d;
    L=sizeof(struct cohort);
    /* assign diameter class to each cohort */
    for (i=0; i<=nco ; i++)
        {
        d=sqrt(sfu[i].tba/0.7854);
        sfu[i].dc=floor((d-5)/2)+1 ;
        };
    /* check each cohort against all subsequent ones looking for a match */
    for (i=0; i<=nco ; i++)
        {
        a=&sfu[i];
        ax=a->gp*100000 + a->model*1000 + a->dc*10 + a->cp;
        if(a->nha >= 0.1)
            for(j=i+1; j<=nco; j++)
                {
                b=&sfu[j];
                bx=b->gp*100000 + b->model*1000 + b->dc*10 + b->cp;
                if(bx==ax)
                    {
                    /* merge cohort b with cohort a. */
                    a->tba=(a->tba * a->nha + b->tba * b->nha)/(a->nha + b->nha);
                    a->nha+=b->nha;
                    b->nha=0.0;
                    };
                };
        };
    /* Remove any cohorts left with N/ha less than 0.01.  Cohorts at the
        i'th position are copied to the j'th position, where j<=i.  j is
        incremented for each non-zero cohort. At the end, nco is adjusted
        to the last value of j.
    */
    for (i=0, j=0; i<=nco ; i++)
        {
        if(sfu[i].nha>0.01)
            {
            /* copy cohort down to new position unless no deleted cohorts */
            if(i>j)
                {
                b=&sfu[j];
                c=&sfu[i];
                memcpy(b,c,L);
                };
            j++;
            };
        };
    nco=j-1;
/* end of MergeCohort routine */
}

InitScreen()
/* sets VGA screen mode and displays CAFOGROM banner */
{
    int xmid, ymid;
    char txt[256];
    initgraph(&gdrv,&gmode,"c:");
    setviewport(XMIN,YMAX,XMAX,YMIN,1);
    xmid=(XMIN+XMAX)/2;
    ymid=(YMIN+YMAX)/2-50;
    setbkcolor(BLUE);
    setcolor(WHITE);
    settextjustify(CENTER_TEXT,BOTTOM_TEXT);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,20);
    moveto(xmid,ymid);
    outtext("CAFOGROM");
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
    moveto(xmid,ymid+30);
    outtext("CPATU Amazon Forest Growth Model");
    settextstyle(SMALL_FONT,HORIZ_DIR,4);
    moveto(xmid,ymid+45);
    strcpy(txt,"Written by Denis Alder under the ");
    strcat(txt,"ODA Silviculture Research Project, December 1994");
    outtext(txt);
}

Setup()
/* reads coefficient arrays from files.  Returns True unless data files
    are corrupted or incompatible. */
{
    FILE *ipf;
    char a; int i, j; float f;
    char eg[3]; int n, m;
    /* basal area increment function coefficients */
    ipf=fopen("TBF.CFD","rt");
    for( i=0; i<NMOD ; i++)
```

```
        {
        fscanf(ipf,"\n%c ",&a);
        modlets[i]=a;
        for( j=0; j<=3 ; j++)
            {
            fscanf(ipf," %f",&f);
            tbf[i][j]=f;
            };
        };
    /* annual mortality coefficients */
    freopen("AMR.CFD","rt",ipf);
    for( i=0; i<NMOD ; i++)
        {
        fscanf(ipf,"\n%c ",&a);
        if(modlets[i]!=a)
            {
            perror("\nError in file AMR.CFD - Models differ from TBF.CFD");
            exit(1);
            };
        for( j=0; j<=3 ; j++)
            {
            fscanf(ipf," %f",&f);
            /* annual mortality % converted to survival per time period */
            psr[i][j]=pow((1-f/100),TSTEP);
            };
            /* max diameter converted to max TBA */
            fscanf(ipf," %f",&f);
            maxtba[i]=f*f*0.7854;
        };
    /* mean ingrowth (recruit) tree diameters */
    freopen("MIT.CFD","rt",ipf);
    for( i=0; i<NMOD ; i++)
        {
        fscanf(ipf,"\n%c ",&a);
        if(modlets[i]!=a)
            {
            perror("\nError in file MIT.CFD - Models differ from TBF.CFD");
            exit(1);
            };
        fscanf(ipf," %f",&f);
        /* recruit mean diameter converted to tree basal area in cm2 */
        mit[i]=f*f*0.7854;
        };
    /* proportion of recruits by guilds, groups, and models */
    freopen("PRG.CFD","rt",ipf);
    for( i=0; ! feof(ipf); i++)
        {
        /* read guild, group, model, proportion */
        fscanf(ipf,"\n%s %d %c %f",&eg, &n, &a, &f);
        /* get index no for guild: Pioneer, Light Demander, Shade Bearer */
        for (j=0; strcmp(eg,guilds[j])!=0 && j<3 ; j++);
        if(j>=3)
            {
            perror("\nUnrecognised guild code in file PRG.CFD");
            exit(1);
            };
        /* get index number for model */
        for(m=0; a!=modlets[m] && m<NMOD ;m++);
        if(m>NMOD)
            {
            perror("\nModel in PRG.CFD not defined in TBF.CFD");
            exit(1);
            };
        /* add details to PRG list */
        prg[i].guild=j;
        prg[i].gp=n;
        prg[i].model=m;
        prg[i].p=f;
        };
    nprg=i-1;
}

GraphFrame()
/* This routine draws the display frame for CAFOGROM.  There are 3 graphs.
    At the top is a histogram of tree numbers over time, shown in 40-year
    sections.  The bottom left gives volume over time.  Bottom right gives
    basal area over time by components. The variables defining the limits
    for these graphs are prefixed u,v,w respectively. */
{
    /* set graphics screen and text style */
    int ntic;
    float xint, yint,xgap,ygap;
    initgraph(&gdrv,&gmode,"c:");
    setviewport(XMIN,YMAX,XMAX,YMIN,1);
    setbkcolor(BLUE);
    Banner();
    settextjustify(LEFT_TEXT,BOTTOM_TEXT);
    settextstyle(SMALL_FONT,HORIZ_DIR,4);
    setcolor(WHITE);
    /* define origins and axis limits of the 3 graphs */
    xint=XMAX-XMIN;
    yint=YMIN-YMAX;
    xgap=0.066*xint;
    ygap=0.066*yint;
    ux0=XMIN+xgap;
    ux1=XMAX-xgap;
    uy0=YSPLIT-ygap;
    uy1=YMAX+ygap;
    uyb=YSPLIT;
    vx0=ux0;
    vx1=XSPLIT-0.5*xgap;
    vy0=YMIN-ygap;
    vy1=YSPLIT+0.5*ygap;
    wx0=XSPLIT+xgap;
    wx1=ux1;
    wy0=vy0;
    wy1=vy1;
    /* diameter distribution histograms */
    DDhist();
    /* Volume/Time graph frame */
    moveto(vx0,vy0);
    ntic=nTtic(Tmax);
    setcolor(WHITE);
    axis(1,(vx1-vx0),ntic,0,Tmax,"Years");
    axis(0,(vy0-vy1),8,0,VOLMAX,"Volume (m3/ha)");
    /* do legend at top left */
    moveto(vx0+10, vy1);
    settextjustify(LEFT_TEXT,BOTTOM_TEXT);
    settextstyle(SMALL_FONT,HORIZ_DIR,4);
    setlinestyle(0,0,NORM_WIDTH);
    line(vx0+5,vy1,vx0+25,vy1);
    outtextxy(vx0+30,vy1,"Commercial");
    setlinestyle(1,0,NORM_WIDTH);
    line(vx0+5,vy1+10,vx0+25,vy1+10);
    outtextxy(vx0+30,vy1+10,"Potential");
    setlinestyle(3,0,NORM_WIDTH);
    line(vx0+5,vy1+20,vx0+25,vy1+20);
    outtextxy(vx0+30,vy1+20,"Non-Commercial");
    setlinestyle(0,0,NORM_WIDTH);
    /* Basal area/Time histogram */
    moveto(wx0,wy0);
    axis(1,(wx1-wx0),ntic,0,Tmax,"Years");
    axis(0,(wy0-wy1),10,0,BAMAX,"Basal area (m2/ha)");
    /* write legend up the side */
    BAlegend();
};

Banner()
/* writes run options at top of screen */
{
    char ntext[10], ttext[60];
    char *title[3]={"Tapajos km67 1981 data (logged 1979)",
                                    "Tapajos km114 1983 data (unlogged)",
                                    "Tapajos km114 1983 data (logged 1982)"};
    setcolor(WHITE);
    settextjustify(LEFT_TEXT,TOP_TEXT);
    settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
    outtextxy(20,1,"CAFOGROM Forest simulation model");
    outtextxy(20,12,title[nData-1]);
}

DDhist()
/* draws the diameter distribution histograms frame */
{
    float vt;
    int i,x,y, nt;
    char numbuf[10];
    setcolor(WHITE);
    /* draw X-axis */
    moveto(ux0,uy0);
    nt=Tmax/(float)TSTEP;
    axis(1,(ux1-ux0),-nt,0,Tmax,"Diameters in 20-cm classes by 5-year periods");
    axis(0,(uy0-uy1),10,0,DDMAX,"Basal area (m2/ha)");
    DDlegend();
}

int EscKey()
/* returns 1 if Esc has been hit, otherwise 0 */
{
    int n; char a;
    if(kbhit())
        {
        a=getch();
        if(a==27)
            {/* Esc key pressed - clear buffer and return with n=1 */
            while(kbhit())getch();
            n=1;
            }
        else
            {/* not an Esc - put it back and return with n=0 */
            ungetch(a);
            n=0;
            };
        }
    else
        n=0;
        return n;
}

void axis(int xy,   int axlen, int nt, float v1, float v2, char *label)
/* draws an axis from the current position of length 'axlen' pixels, with 'nt'
        tic marks, labelled from v1 to v2, and text 'label'.  If 'xy' is 0
        an x-axis drawn, otherwise a y-axis.  If 'nt' is negative, the label
        stops one tic short of the axis end. */
{
float vt, vi, ticint;
int x, x0, y0, x1, y, y1, maxch;
int i;
char numbuf[20];
x0=getx();
y0=gety();
/* draw axis line */
if(xy) {x1=x0+axlen; y1=y0;} else { x1=x0 ; y1=y0-axlen;};
lineto(x1,y1);
/* tic mark and scale intervals */
if(nt<0) ticint=(float)axlen/(abs(nt)+1) ; else ticint=(float)axlen/nt;
vi=(v2-v1)/abs(nt);
moveto(x0,y0);
vt=v1;
if(xy)
        {
        /* tick and label horizontal axis */
        settextjustify(CENTER_TEXT, TOP_TEXT);
        settextstyle(SMALL_FONT, HORIZ_DIR,4);
        for ( i=0, x=x0 ; i< =abs(nt) ; i+ +)
                {
                lineto(x,y0+4);
                itoa((int)vt, numbuf, 10);
                outtext(numbuf);
                vt+ =vi;
                x=x0+(i+1)*ticint;
                moveto(x,y0);
                };
        /* write text centred along axis */
        moveto(x0+axlen/2,y0+20);
        outtext(label);
        }
else
        {
        /* tick and label and Y axis */
        settextjustify(RIGHT_TEXT, CENTER_TEXT);
        settextstyle(SMALL_FONT, HORIZ_DIR,4);
        maxch=0;
        for ( i=0, y=y0 ; i< =abs(nt) ; i+ +)
                {
                lineto(x0-4,y);
                itoa((int)vt, numbuf, 10);
                outtext(numbuf);
                maxch=(strlen(numbuf)>maxch  ? strlen(numbuf) : maxch);
                vt+ =vi;
                y=y0-(i+1)*ticint;
                moveto(x0,y);
                };
        /* write text centred along axis */
        settextjustify(RIGHT_TEXT, CENTER_TEXT);
        settextstyle(SMALL_FONT, VERT_DIR,4);
        moveto(x0-8*(1+maxch),y0-axlen/2);
        outtext(label);
        };
    /* return to initial position */
    moveto(x0,y0);
    }

    int nTtic(int tmax)
    /* works out number of tic marks required for time axis, based on
        algorithm of dividing by 5, 10, 15, ... until an exact dividend
        results between 2 and 8 */
    {
        int k, n, r;
        for (k=5; k<=tmax/2 ; k+=5)
            {
            n=tmax/k;
            r=tmax % k;
         if (r==0 && n>=2 && n<=8) return n;
            };
        /* modify Tmax so result is not prime */
        Tmax+=5;
        return nTtic(Tmax);
    }

    GraphOut()
    /* calculates summaries from cohort array SFU in each time period and
        presents them graphically */
    {
        DDhistOut();
        VolGraphOut();
        if(Tsim>0)BAGraphOut();
    };

    VolGraphOut()
    {
        static float LastVol[3]; static int LastX;
        float Vol[3], f;
        int i,k, g, x1, x2,y1,y2,xt, yt, offset;
        char annotate[25];
        struct cohort *co;
        offset=0;
        /* calculates volume over 45-cm for (1) Commercial, (2) Commercial +
            potential species, and (3) All species.  These are drawn as a
            line from the previous point.  At Tsim=0, no line is drawn, only
            the first point is set. */
        /* calculate volumes */
        for (k=0; k<=2 ; k++) Vol[k]=0;
        for (i=0; i<nco ; i++)
            {
            /* pointer to this cohort */
            co=&sfu[i];
            g=co->gp;
            /* utilization group - Commercial (1-2), Potential (3-4), Other (5-9) */
            k=(g<=2 ? 0 :( g<=4 ? 1 : 2));
            Vol[k]+=voleqn(co->tba)*co->nha;
            };
        /* if Tsim more than zero draw a line from volumes at last period */
        if(Tsim>0)
            {
            x1=LastX;
            f=Tsim/(float)Tmax;
            x2=vx0+f*(vx1-vx0);
            yt=vy0;
            for (k=0; k<=2; k++)
            {
            f=LastVol[k]/VOLMAX;
            y1=vy0+f*(vy1-vy0);
            f=Vol[k]/VOLMAX;
            y2=vy0+f*(vy1-vy0);
            setlinestyle(k,0,NORM_WIDTH);
            line(x1,y1,x2,y2);
            yt=smallest(y1,y2,yt);
            };
        /* annotate if thinning or logging */
        settextjustify(CENTER_TEXT, BOTTOM_TEXT);
        settextstyle(SMALL_FONT, VERT_DIR,4);
        xt=(x1+x2)/2;
        yt-=10;
        if(LogFlag)
            {
            sprintf(annotate,"Log %4.1fm3",hVol);
            outtextxy(xt,yt,annotate);
            LogFlag=0;
            hVol=0;
            offset=12;
            };
        if(ThinFlag)
            {
            sprintf(annotate,"Thin %4.1fm2",thBA);
            outtextxy(xt+offset,yt,annotate);
            ThinFlag=0;
            };
        }
    else x2=vx0;
    /* save current point positions for next period */
    LastX=x2;
    for (k=0; k<=2; k++) LastVol[k]=Vol[k];
}

float voleqn(float tba)
/* Natalino's Tapajos volume equations */
{
    float d,v;
    d=sqrt(tba/0.7854);
    if (d>=45)
        v=exp(-7.6281+2.1809*log(d));   /* trees over 45 cm dbh */
    else
        if(d>=15)
            v=-0.0994+0.00091941*d*d;   /* trees 15-45 cm dbh */
        else
            v=0.0;
    return v;
}

int smallest(int a, int b, int c)
/* returns the smallest of a,b, or c */
{
    if (a<b)
        if (a<c)
            return a;
        else
            return c;
    else
        if (b<c)
            return b;
        else
```

```
                return c;
}

DDhistOut()
/* draws diameter distribution histogram, 1 bar per 20-cm class, grouped
    by guilds: shade bearer, light demander, pioneer */
{
    float DDbars[3][10], gwidth;
    int nbytes, k, i, j, g, yL;
    struct cohort *co;
    int BarCol[]={WHITE, WHITE, WHITE};
    int BarPat[]={SOLID_FILL, INTERLEAVE_FILL, EMPTY_FILL};
    int ng, bwidth, bx0,bx00, bx1,by0,by1;
    for (k=0; k<=2; k++) for (j=0; j<=10; j++) DDbars[k][j]=0.0;
    /* total diam distribution from cohort array */
    for(i=0; i<nco; i++)
        {
        co=&sfu[i];
        /* trees over 145 cm (which will have very low stocking) are ignored */
        j=floor((sqrt(co->tba/0.7854)-5.0)/20);
        j= (j>9 ? j=9 : j);
        switch(co->gp)
            {
            case 7:   k=0;            /* pioneers */
                        break;
            case 2:  ;
            case 4:  k=1;             /* light demanders */
                        break;
            default:   k=2;               /* shade bearers */
            };
        DDbars[k][j]+=co->tba*co->nha*0.0001;
        };
    /* plot bar data, 1 line per bar */
    ng=Tmax/(float)TSTEP+1;
    gwidth=(ux1-ux0)/(float)ng;
    bwidth=gwidth/8;
    bx00=ux0+ gwidth*Tsim/TSTEP;
    setcolor(WHITE);
    for (j=0; j<7; j++)
        {
        bx0=bx00+j*bwidth;
        bx1=bx0+bwidth;
        by0=uy0;
        for(k=0; k<3 ;k++)
            {
            yL=(DDbars[k][j]/DDMAX)*(uy0-uy1);
            by1=by0-yL;
            setfillstyle(BarPat[k], BarCol[k]);
            bar(bx0,by1, bx1,by0);
            rectangle(bx0,by1,bx1,by0);
            by0=by1;
            };
        };
};
```

```
BAGraphOut()
/* Draws the bars of the basal area/time diagram.  These show standing
    basal area (shaded, brown), increment (shaded, yellow), recruitment
    (shaded, cyan), and mortality (open, magenta). */
{
    int i,bx0,bx1,bwidth,by0,by1;
    float yf;
    /* components of BA are stored in this array to facilitate plotting */
    float ba[7];
    /* colours and patterns for sba, sbai, rba, mba */
    int BarPat[]={SOLID_FILL, XHATCH_FILL, INTERLEAVE_FILL,SLASH_FILL, CLOSE_DOT_FILL,
                                WIDE_DOT_FILL,EMPTY_FILL};
    /* locate the left and right coordinates for the current bar */
    bwidth=(wx1-wx0)*((float)TSTEP/(float)Tmax)*0.8;
    bx0=wx0+(Tsim/(float)Tmax)*(wx1-wx0)-bwidth/2;
    bx1=bx0+bwidth;
    /* y-axis scale factor, pixels per unit of BA */
    yf=(wy0-wy1)/(float)BAMAX;
    /* store basal area components in an array */
    ba[0]=hBA;
    ba[1]=lostBA;
    ba[2]=sba0-lostBA;
    ba[3]=thBA;
    ba[4]=sbai;
    ba[5]=rba;
    ba[6]=mba;
    lostBA=0.0;
    /* draw bars */
    setlinestyle(SOLID_LINE,0, NORM_WIDTH);
    setcolor(WHITE);
    by0=wy0;
    for (i=0; i<=6 ; i++)
        if(ba[i]>0)
            {
            setfillstyle(BarPat[i], WHITE);
            by1=by0-ba[i]*yf;
            bar(bx0, by1, bx1, by0);
            rectangle(bx0, by1, bx1, by0);
            by0=by1;
            };
}

BAlegend()
/* writes the legend for the BA graph */
{
    int i,x1,x2,y1,y2;
    int BarPat[]={SOLID_FILL, XHATCH_FILL, INTERLEAVE_FILL,SLASH_FILL, CLOSE_DOT_FILL,
                                WIDE_DOT_FILL,EMPTY_FILL};
    char*Labels[]={"Harvested","Logdamage","Standing","Thinned","Increment","Recruitment","Mortality"};
    setcolor(WHITE);
    settextjustify(LEFT_TEXT, BOTTOM_TEXT);
    settextstyle(SMALL_FONT, HORIZ_DIR,4);
    y2=wy1-8;     y1=y2+8;      x1=wx0+3; x2=x1+12;
    for (i=0; i<=6 ; i++)
        {
        setfillstyle(BarPat[i], WHITE);
        bar(x1, y2, x2, y1);
        rectangle(x1, y2, x2, y1);
        outtextxy(x2+3,y1,Labels[i]);
        /* move to next line, or start a new column if i=2 */
        if(i==3)  { x1+=90; x2+=90; y2=wy1-8; y1=y2+8;}
            else { y1+=12;     y2+=12;};
        };
}

DDlegend()
/* draws legend to DD histogram */
{
    int BarPat[]={SOLID_FILL, INTERLEAVE_FILL, EMPTY_FILL};
    int i,x1,x2,y1,y2;
    char *Labels[]={"Pioneer","Intolerant","Tolerant"};
    setcolor(WHITE);
    settextjustify(LEFT_TEXT, BOTTOM_TEXT);
    settextstyle(SMALL_FONT, HORIZ_DIR,4);
    y2=6;    y1=y2+8;     x1=wx1-50; x2=x1+12;
    for (i=0; i<=2 ; i++)
        {
        setfillstyle(BarPat[i], WHITE);
        bar(x1, y2, x2, y1);
        rectangle(x1, y2, x2, y1);
        outtextxy(x2+3,y1,Labels[i]);
        y1+=12;
        y2+=12;
        };
}
```

INDEX