Computational Methods for Data Analysis

JOHN M. CHAMBERS
Bell Laboratories
Murray Hill, New Jersey

JOHN WILEY & SONS, New York • Chichester • Brisbane • Toronto

Published by John Wiley & Sons, Inc.

Library of Congress Cataloging in Publication Data

Chambers, John M
Computational methods for data analysis.

(Wiley series in probability and mathematical statistics)
Includes index.
1. Mathematical statistics—Data processing.
2. Numerical analysis—Data processing. I. Title.

QA276.4.C48 519.4 77-9493
ISBN 0-471-02772-3

Printed in the United States of America

10 9 8 7 6 5 4 3 2

PREFACE

The material in this book covers the major computational methods which are important for data analysis. The approach is to present the essential results on each topic, including an appraisal of currently competitive methods and references to selected algorithms. A special Appendix gathers together these references, with information on the language and format of the algorithms and how to obtain them. Most chapters conclude with a set of problems. These are intended to be of practical value to the reader. None is entirely trivial and some involve designing a significant set of algorithms.

Readers looking for a specific topic should go first to the quick reference on the inside back cover. This will give page references for both the method and the algorithms. If the topic of interest is not obviously covered in the quick reference, try the index.

This is the first book to attempt a complete picture of computing for data analysis. I hope the result will be useful for several groups of people. Those involved in analysing data may look here to find computing methods currently competitive for specific problems. Users of existing packages and programs may compare the reliability, accuracy and cost of the methods used to the current best (and, I hope, will react strongly to any defects they find in the packages).

Those who are or would like to be active in statistical computing will find introductions to various topics, a wealth of unsolved or partially solved problems, and references to more detailed treatments. (Some of the important areas for future work are listed in the Index, under "open problems".) Finally, I hope professionals in the various fields of computing science will find it useful to see the viewpoint and needs suggested by data analysis.

Computing for data analysis is an important, challenging and varied field. Data analysis is increasingly an important part of scientific research, economic analysis and many other activities. Major analyses usually require extensive computing; hence, the validity and adequacy of the computation must be demonstrable to

support the analysis. An awareness of the best computing methods and of their limitations will be increasingly important. It is essential, when evaluating computing techniques, to be aware as well of the real goals of the data analysis, so as to sort out the relevant from the merely interesting. Having invoked these important responsibilities, I should quickly add that working on these frontiers of computing and data analysis is frequently exhilarating and great fun.

I have had the opportunity of working at one time or another in all the major areas described in this book. As a result, any list of indebtedness would be sure to offend by omission. I can only hope that the many experts who have given generously of their time and advice will detect some beneficial effects in the result. The support of Bell Laboratories in the research and writing of this book is gratefully acknowledged. I am particularly grateful for the stimulation of discussions with many colleagues there, past and present. The entire contents of the book have been set on a computer photocomposition system at Bell Laboratories. It is hoped that this support will both add to the timeliness of the material and allow some cost savings to be passed on to buyers. Portions of the material have been presented in courses at Harvard University, Princeton University and Bell Laboratories. Comments and questions from students have been valuable in subsequent revisions.

Note to the Reader

The Chapters of the book are as self-contained as reasonably possible. References to Equations, Figures and Sections within the Chapter are given without Chapter number. The References are collected, by Chapter, beginning on page 228. The Appendix deals with algorithms by Chapter.

CONTENTS

CHAPTER ONE

Introduction

a. General Approach

This book is intended to assist those who are involved in the planning, selection, or development of computational support for data analysis. There is general recognition of the importance of such support for effective research and development in all branches of science. The ability to do massive amounts of programmed computations is the foremost change in the intellectual environment in the second half of the twentieth century. The effective amount of calculation possible in a given time has increased by about five orders of magnitude. Several revolutionary changes in equipment and, to a somewhat lesser extent, in programming facilities have entirely altered the scope of computing. The capabilities and practical economics of computing continue to change at a rapid rate in many respects.

Chapters 2 through 8 treat major topics of computational technique. Chapter 2 discusses programming and program evaluation in general. Chapter 3 deals with data management and related problems. Chapters 4 through 7 cover numerical methods (general methods, linear methods, nonlinear fitting, and simulation). Chapter 8 discusses graphical computation.

A wide variety of computational topics is treated, since data analysis needs support from nearly every branch of computer science. The simpler approach of concentrating on a few, relatively straightforward topics will not provide sufficient guidance for practical work. The computational topics are treated from the special viewpoint of data analysis. The emphasis, techniques, and recommendations, therefore, differ significantly from those for other applications.

For each topic, the discussion identifies the essential computational problems, outlines and compares the currently competitive approaches and provides references to more detailed material, including reliable published or generally available algorithms where

possible. For the most part, it is assumed that some programming effort will be required, either in implementing procedures or in modifying existing algorithms or other special-purpose software.

No attempt is made to rate existing packages or languages for statistical computing in any overall sense. Users will be able to check the facilities provided against the currently competitive methods for a particular problem. Also, the steps required to extend or improve the current capabilities of the package can be assessed from the descriptions here. The concern is to outline what is currently known and available to solve specific problems. Users of packages may assess the facilities provided against this background. We especially try to suggest how new or extended facilities can be developed.

Both current abilities and long-range prospects vary greatly among different areas of computing. Some problems are relatively easy and can be considered to be understood, in broad outline at least; for example, linear least-squares and sorting. Others are in a less satisfactory state at the moment, but can in principle be well handled: for example, provision of language interfaces and graphical computations. Finally, some areas are intrinsically too complex ever to have single, general solutions. For such areas, a vague general problem must be sharpened to make it computationally meaningful. Examples are the simulation of random numbers and, possibly, general fitting of nonlinear models.

b. Data Analysis and Computing

As a general preliminary to detailed discussions, it is useful to characterize possible computing environments and the kinds of data analysis likely to arise. Such questions affect the feasibility or relevance of techniques to be discussed.

The data analysis may be characterized by such questions as:

(1) Is it primarily routine (consisting of specialized and repetitive analyses) or primarily exploratory (unpredictable as to the kind of problem and the appropriate response)?

(2) Are the problems large or small, both in terms of the volume of data and of the resources (personnel, computing facilities, etc.) available?

Exploratory analysis places greater emphasis on the flexibility and extensibility of the computing facilities. Routine analysis allows

greater emphasis on efficiency and ease of standard use of programs, and permits greater effort to be made in refining computing procedures. Generally, routine analysis favors specialized systems, while exploratory analysis favors general, open-ended support.

Small projects will not be as likely to undertake major new programming efforts. The primary emphasis will be on quick and easy access to reasonably appropriate computing tools. At the same time, direct checking of results and adhoc revisions and interpretation will be more feasible on small bodies of data. Larger projects may not be justified in compromising the details of their analysis or their computations, while large data sets always require great care in checking for errors and misinterpretations. Greater control over the details of computation will then be required.

A different characteristic of the environment is the general style in which computing for data analysis is presented. In Section **2.b** some criteria for evaluating individual algorithms and systems are discussed. A more general characteristic is the contrast between special-purpose statistical systems and general-purpose, nonstatistical languages. The merits of these have been the focus of considerable debate. The two extremes are a statistical system that replaces all programming in general languages and direct programming in a general language. Intermediate forms include statistical systems that permit the kind of programming done in ordinary languages, and programming in general languages with extensive support from subroutine libraries and other facilities. Special-purpose systems can provide commonly used analyses or data structures as simple primitives; initial and routine use is facilitated. General-purpose approaches have advantages of extensibility and flexibility. In particular, the incorporation of new procedures written elsewhere is usually more straightforward.

In assessing various approaches to computing for data analysis, it is relevant to consider changes in the form of user communication with computers and changes in the population of potential users since initial development of statistical systems. In the typical computer facility until 1965, user communication was by an externally prepared program, usually on punched cards, presented for a relatively slow run (with several hours or more elapsing before obtaining results). Such an environment discourages innovative, flexible use of computer systems. Subsequent developments have led to computer access via interactive terminals with essential support of text editors and other non-numerical systems. Statistical

computations also typically are executed more rapidly as well, either in an interactive system, or by a spawned job with a short time to completion. The new style encourages direct and innovative participation by data analysts in computing. A contemporary development is the training of a generation of data analysts who had early exposure to computing and who accept computer programming as a natural expression of the analysis they intend to perform. These developments are relevant to planning because better user communication and data analysts who are at ease with programming allow, and often demand, broader and more flexible computer support. Some restrictive package systems, for example, were suitable for the earlier environment but needed to be revised to be acceptable subsequently.

For future planning, however, the best features of both approaches should be sought. Building upon a widely available general programming language (probably FORTRAN), and taking advantage of high-quality existing algorithms, will provide a strong problem-solving support. Convenient user interfaces, where needed, may be built upon this basis, using modern techniques of language design. (See Sections **2.d** to **2.f**.)

CHAPTER TWO

Programming

This chapter considers some questions about programs that apply across most areas of technique. What good features should programs have? How can competing algorithms be compared? What principles underlie the design and writing of good programs? What features of programming languages are relevant to computing for data analysis? In the following sections, a few points are discussed that relate to these questions.

a. Structure of Computations in Data Analysis

The process of data analysis, whether or not it is carried out by computer, has a basic structure that closely parallels the operation of a computer program. The sequence of steps in each case involves the same four components: the acquisition of external data (input), the planning and definition of the analysis (programming), the execution of the analysis (calculation), and the display of results and summaries (output).

These components define the areas that must be considered when developing a computational capability for data analysis. At the same time, they are each areas in which much development of computing technology and methods has occurred. The provision of effective data analysis requires some familiarity with recent developments in these areas. Later chapters present the important current techniques in various specialized areas of computing. The present chapter attempts to take a general look at the process of providing computer programs.

General advice on the planning and implementation of statistical computing facilities should be given with modesty and caution. Each organization differs in the goals and constraints applied to its data analysis, in the relative priorities it assigns to these, and in the time-scale on which these goals and constraints are to be considered.

At the same time, the resources available will also differ, in terms of the number and skills of personnel, the quantity and scope of computing hardware, and the extent and usefulness of existing programs.

Also, the computational requirements for data analysis differ from, but still overlap with, the needs of other computer applications. Therefore, an approach that treats data analysis as if it were the sole user of a computer system may overlook the benefits of planned cooperation with other groups of computer users.

The approach to computers will certainly depend on all these local conditions. However, there are useful general concepts to be applied in assessing each particular case. By keeping such concepts in mind, planners may achieve a more relevant and adequate facility within their environment.

The first principle to establish is that the computational needs are part of the whole process of data analysis. The effectiveness of a particular computing tool, therefore, lies in its overall role in the analysis to which it is applied. The costs and benefits must be balanced in this broader context, not just in terms of the computing itself. This principle, apparently self-evident, is quite difficult to apply consistently in practice, as is shown by many of the examples cited in later chapters.

The next questions to answer are then: "What good things can computers provide in data analysis?" and "What difficulties and costs can be associated with computing?" It is possible to give some general, although incomplete, answers to these questions.

The *benefits* of good computing support for data analysis are of great magnitude. While most of them (increased speed of computation, ability to handle larger data bases, the availability of new and better algorithms) may be regarded as quantitative, they are often sufficiently major to revolutionize the entire approach to data analysis. In many applications, however, the full benefit of modern computing methods will be obtained only when the traditional statistical methodology and concepts are rethought. Frequently, the result is a more natural and unrestricted attitude to the methods themselves, uninhibited by limitations that were originally imposed by hand-calculation methods, and later came to be considered intrinsic to the methods themselves. A typical specific example is the

conventional assumption of full rank (nonsingularity) in statistical procedures such as linear regression, and the resulting problems due to near-singularity (multi-collinearity). Once the linear model is described in the natural, general framework of orthogonal bases, the *computational* problems of general rank in linear models are by no means insuperable. To exploit the computational methods fully, however, one must consider the *statistical* implications of such questions as uncertain rank (see Section **5.f**).

The *costs* of computing are more frequently discussed, at least in the narrow sense of computer time or other quantifiable resources. Section **b** gives some general comments on this topic, and the remainder of the book contains discussions of the cost of specific procedures. The extent to which such costs ought to be the *main* concern in data analysis is questionable, however. For very large, routine projects, computing efficiency may be the determining factor. For other situations, particularly in exploratory work, the other criteria discussed in Section **b** are often of greater importance.

b. Program Evaluation

The evaluation of any set of proposed computing tools can be organized around four main headings:

1. usefulness,
2. reliability,
3. cost.
4. convenience.

Each of these provide a framework for planning and assement.

Usefulness. The most important requirement of a program or computing system is that it solve a relevant and important set of problems. If a particular set of problems provided the incentive for the programs, one must first look at this local question of usefulness. The evaluation will require some care. For example, does the program *fully* solve the problem, including provision of data acquisition and display of results where necessary? The new program need not do all this itself, but if it does not, communication between it and other programs for this purpose must be straightforward, and if the other programs do not already exist, the cost of writing them must be included.

If the program was written for a continuing series of problems, it will normally be designed around only the few examples currently available (and frequently only a subset of these). Planners need to consider the scope of the *full* series before restricting the applicability of the program design. If programmer, designer and data analyst are different individuals, it is particularly useful that the first two feed back their understanding of the program's scope to the data analyst *before* writing the program.

The broader aspects of program usefulness are sometimes overlooked. The question here is whether the program will be easily useful with, at most, straightforward modifications for other problems. Is the program designed so that it can be used for related but distinct problems? Is it possible to understand the program and modify it to apply to different problems? Can it be moved reasonably easily to a different environment, possibly on a different computer system?

The reason for emphasizing this consideration is that real gains can be made here. It is frequently possible to solve an immediate problem with little extra effort (sometimes, in fact, with less effort) in such a way that the general, long-term value of the programs is greatly increased. To achieve this the larger context must be kept in mind from an early stage of the planning. The programmer deeply involved in details is less likely to see this context than an overall planner slightly removed from the details. Where planner and programmer are the same person, this suggests that a conscious effort should be made occasionally to step back from the immediate problem and consider the overall strategy. Sections **c** and **d** discuss several aspects of increasing program usefulness.

Reliability. The reliability or accuracy of computations and the cost of alternative methods are more technical questions that can only be answered relative to the current understanding of particular computing problems. Estimation of reliability is a central theme of numerical analysis, which is discussed in Chapter 4 and elsewhere. A closely related concept is *sensitivity,* the change in the result of a computation as a result of a change in the data provided to it. Where data is uncertain, as nearly all observational data must be, such analysis must become particularly important. Except for a few relatively simple techniques, sensitivity analysis is not yet a well-developed tool in scientific data analysis.

Different questions of reliability arise with respect to the integrity of the data and the programs themselves. Statistical techniques for detecting gross errors in data or for reducing their effect on subsequent calculations have been developed for some of the standard analyses (se Section **5.j**). Computational procedures designed to ensure the integrity of data have been included in many data base management systems.

Cost. Cost estimates and cost comparisons have been developed for many algorithms and is discussed frequently in later chapters. There are two basic approaches to cost evaluation: theoretical and empirical. The most desirable measures of cost would seem to be *actual* cost, and for the more complicated procedures no other measures may be available. The relevant cost measures will depend on the machine and the viewpoint of the user. On a small or wholly dedicated machine with a single user the natural measure is the elapsed time required to complete the job. Even in so simple an environment, some qualifications are necessary. The true cost of lengthy computations may be substantially lessened if they do not require constant supervision. On small computers that are normally idle at night, users are often ingenious at inventing computing tasks that can be started and left to run indefinitely. Such programs, it may be argued, run at very little cost, however long they take. (Multiprogram systems, applying similar logic, may provide a low cost, background, or deferred grade of service, which runs essentially by use of otherwise idle resources.)

On modern multiprogrammed systems, cost measures are more indirect. The supplier of the computing facilities wishes to obtain the most computing from the available facilities, by ensuring an efficient allocation of resources among users. In a commercially organized computing facility, the supplier will generally charge users for their consumption of the various resources within the system (central processor time, peripheral storage, input/output, printing, etc.). In a purely commercial environment, this will be done so as to realize a suitable return to the supplier. In a semicommercial situation, as in a university or scientific establishment that runs its own computer center and charges others in the organization for computing, the goals are usually to obtain efficient use of the computer and, perhaps, to distribute the cost of supporting the computer

facility among members of the organization in proportion to their real use of it. These latter goals may conflict and the usual marketplace approach to pricing computer facilities does not automatically guarantee efficiency in the use of the facility. Various alternative schemes have been suggested for in-house computing allocation. There is unlikely to be an ideal scheme for all circumstances, but a key ingredient would seem to be responsiveness to actual usage and an awareness of current or potential blockages in the flow of user programs.

The user's strategy in a market or semimarket environment generally is to obtain as much of the computing he wants as possible given his budget for computing resources. As a result, the cost of alternative computational procedures or systems must be evaluated in terms of the totality of resources required. A larger program that executes faster may or may not be preferrable; similarly, the choice between data management that uses more peripheral storage and one that requires more frequent access to that storage will depend on current pricing policies. This is part of the rub from the user's viewpoint. The flexibility in pricing that the supplier sometimes needs to respond to varying demand may make it difficult for an individual user to develop a stable efficient algorithm for some of his major computations.

The topics we have just discussed have become significant technical problems with the development of large, complex, multiuser computing systems. Simply measuring and understanding the performance of such systems and analyzing the results of changes in the systems or their pricing strategy may require sophisticated data analysis. Such techniques have been labeled *compumetrics*. In fact, complex computing systems provide interesting challenges for data analysis: the data is abundant and significant relationships and patterns are likely to exist, but the appropriate models and estimates are frequently not obvious.

In addition to the ambiguities and instabilities mentioned, empirical cost comparisons are obviously defined only for a particular machine and frequently change when the same procedure is used with a different mixture of other programs, particularly in a multiuser environment. Designing or selecting algorithms for multicomputer use requires some measures of cost that are valid generally. For this purpose, theoretical cost calculations may sometimes