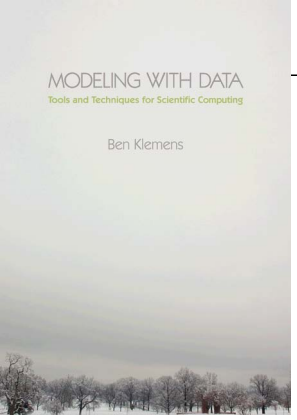

MODELING WITH DATA
Tools and Techniques for Scientific Computing
Ben Klemens

Modeling with Data

Modeling with Data

Tools and Techniques for Scientific Computing

Ben Klemens

PRINCETON UNIVERSITY PRESS
PRINCETON AND OXFORD

Published by Princeton University Press
41 William Street, Princeton, New Jersey 08540

In the United Kingdom: Princeton University Press
6 Oxford Street, Woodstock, Oxfordshire, OX20 1TW

Klemens, Ben.
Modeling with data : tools and techniques for scientific computing / Ben Klemens.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-691-13314-0 (hardcover : alk. paper)
1. Mathematical statistics. 2. Mathematical models. I. Title.
QA276.K546 2009
519.5–dc22 2008028341

British Library Cataloging-in-Publication Data is available

This book has been composed in LaTeX

The publisher would like to acknowledge the author of this volume for providing the camera-ready copy from which this book was printed.

Printed on acid-free paper. ∞

press.princeton.edu

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

We believe that no one should be deprived of books for any reason.

—Russell Wattenberg, founder of the Book Thing

The author pledges to donate 25% of his royalties to the Book Thing of Baltimore, a non-profit that gives books to schools, students, libraries, and other readers of all kinds.

CONTENTS

PREFACE

> Mathematics provides a framework for dealing precisely with notions of "what is."
> Computation provides a framework for dealing precisely with notions of "how to."
>
> —Alan J Perlis, in Abelson et al. [1985, p xvi]

SHOULD YOU USE THIS BOOK? This book is intended to be a complement to the standard stats textbook, in three ways.

First, descriptive and inferential statistics are kept separate beginning with the first sentence of the first chapter. I believe that the fusing of the two is the number one cause of confusion among statistics students.

Once descriptive modeling is given its own space, and models do not necessarily have to be just preparation for a test, the options blossom. There are myriad ways to convert a subjective understanding of the world into a mathematical model, including simulations, models like the Bernoulli/Poisson distributions from traditional probability theory, ordinary least squares, simulations, and who knows what else.

If those options aren't enough, simple models can be combined to form multilevel models to describe situations of arbitrary complexity. That is, the basic linear model or the Bernoulli/Poisson models may seem too simple for many situations, but they are building blocks that let us produce more descriptive models. The overall approach concludes with multilevel models as in, e.g., Eliason [1993], Pawitan [2001] or Gelman and Hill [2007].

Second, many stats texts aim to be as complete as possible, because completeness and a thick spine give the impression of value-for-money: you get a textbook *and* a reference book, so everything you need is guaranteed to be in there somewhere.

But it's hard to learn from a reference book. So I have made a solid effort to provide a narrative to the important points about statistics, even though that directly implies that this book is incomplete relative to the more encyclopædic texts. For example, moment generating functions are an interesting narrative on their own, but they are tangential to the story here, so I do not mention them.

Computation The third manner in which this book complements the traditional stats textbook is that it acknowledges that if you are working with data full-time, then you are working on a computer full time. The better you understand computing, the more you will be able to do with your data, and the faster you will be able to do it.

People like to characterize computing as fast-paced and ever-changing, but much of that is just churn on the syntactic surface. The fundamental concepts, conceived by mathematicians with an eye toward the simplicity and elegance of pencil-and-paper math, have been around for as long as anybody can remember. Time spent learning those fundamentals will pay off no matter what exciting new language everybody happens to be using this month.

I spent much of my life ignoring the fundamentals of computing and just hacking together projects using the package or language of the month: C++, Mathematica, Octave, Perl, Python, Java, Scheme, S-PLUS, Stata, R, and probably a few others that I've forgotten. Albee [1960, p 30] explains that "sometimes it's necessary to go a long distance out of the way in order to come back a short distance correctly;" this is the distance I've gone to arrive at writing a book on data-oriented computing using a general and basic computing language. For the purpose of modeling with data, I have found C to be an easier and more pleasant language than the purpose-built alternatives—especially after I worked out that I could ignore much of the advice from books written in the 1980s and apply the techniques I learned from the scripting languages.

The politics of software

All of the software in this book is *free software*, meaning that it may be freely downloaded and distributed. This is because the book focuses on portability and replicability, and if you need to purchase a license every time you switch computers, then the code is not portable.

If you redistribute a functioning program that you wrote based on the GSL or Apophenia, then you need to redistribute both the compiled final program and the source code you used to write the program. If you are publishing an academic work, you should be doing this anyway. If you are in a situation where you will distribute only the output of an analysis, there are no obligations at all.

This book is also reliant on *POSIX*-compliant systems, because such systems were built from the ground up for writing and running replicable and portable projects. This does not exclude any current operating system (OS): current members of the Microsoft Windows family of OSes claim POSIX compliance, as do all OSes ending in X (Mac OS X, Linux, UNIX, . . .).

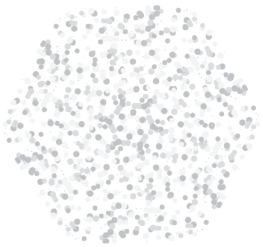
WHAT IS THE LEVEL OF THIS BOOK? The short answer is that this is intended for the graduate student or independent researcher, either as a supplement to a standard first-year stats text or for later study. Here are a few more ways to answer that question:

- *Ease of use versus ease of initial use*: The majority of statistics students are just trying to slog through their department's stats requirement so they can never look at another data set again. If that is you, then your sole concern is ease of initial use, and you want a stats package and a textbook that focus less on full proficiency and more on immediate intuition.[1]
  Conversely, this book is not really about solving today's problem as quickly as physically possible, but about getting a better understanding of data handling, computing, and statistics. Ease of long-term use will follow therefrom.
- *Level of computing abstraction*: This book takes the fundamentals of computing seriously, but it is not about reinventing the wheels of statistical computing. For example, *Numerical Recipes in C* [Press et al., 1988] is a classic text describing the algorithms for seeking optima, efficiently calculating determinants, and making random draws from a Normal distribution. Being such a classic, there are many packages that implement algorithms on its level, and this book will build upon those packages rather than replicate their effort.
- *Computing experience*: You may have never taken a computer science course, but do have some experience in both the basics of dealing with a computer and in writing scripts in either a stats package or a scripting language like Perl or Python.
- *Computational detail*: This book includes about 90 working sample programs. Code clarifies everything: English text may have a few ambiguities, but all the details have to be in place for a program to execute correctly. Also, code rewards the curious, because readers can explore the data, find out to what changes a procedure is robust, and otherwise productively break the code.
  That means that this book is not computing-system-agnostic. I have made every effort to make this book useful to people who have decided to not use C, the GSL, or Apophenia, which means that the more tedious details of these various systems are left to online manuals. But I have tried to include at least enough about the quirks of these various systems so that you can understand and modify the examples.
- *Linear algebra*: You are reasonably familiar with linear algebra, such that an expression like $\mathbf{X}^{-1}$ is not foreign to you. There are a countably infinite number of linear algebra tutorials in books, stats text appendices, and online, so this book does not include yet another.
- *Statistical topics*: The book's statistical topics are not particularly advanced or trendy: OLS, maximum likelihood, or bootstrapping are all staples of first-year grad-level stats. But by creatively combining building blocks such as these, you will be able to model data and situations of arbitrary complexity.

[1] I myself learned a few things from the excellently written narrative in Gonick and Smith [1994].

Modeling with Data

1



STATISTICS IN THE MODERN DAY

> Retake the falling snow: each drifting flake
> Shapeless and slow, unsteady and opaque,
> A dull dark white against the day's pale white
> And abstract larches in the neutral light.
>
> —Nabokov [1962, lines 13–16]

Statistical analysis has two goals, which directly conflict. The first is to find patterns in static: given the infinite number of variables that one could observe, how can one discover the relations and patterns that make human sense? The second goal is a fight against *apophenia*, the human tendency to invent patterns in random static. Given that someone has found a pattern regarding a handful of variables, how can one verify that it is not just the product of a lucky draw or an overactive imagination?

Or, consider the complementary dichotomy of objective versus subjective. The objective side is often called *probability*; e.g., given the assumptions of the Central Limit Theorem, its conclusion is true with mathematical certainty. The subjective side is often called *statistics*; e.g., our claim that observed quantity A is a linear function of observed quantity B may be very useful, but Nature has no interest in it.

This book is about writing down subjective models based on our human understanding of how the world works, but which are heavily advised by objective information, including both mathematical theorems and observed data.[1]

[1] Of course, human-gathered data is never perfectly objective, but we all try our best to make it so.

The typical scheme begins by proposing a model of the world, then estimating the parameters of the model using the observed data, and then evaluating the fit of the model. This scheme includes both a descriptive step (describing a pattern) and an inferential step (testing whether there are indications that the pattern is valid). It begins with a subjective model, but is heavily advised by objective data.

Figure 1.1 shows a model in flowchart form. First, the descriptive step: data and parameters are fed into a function—which may be as simple as *a is correlated to b*, or may be a complex set of interrelations—and the function spits out some output. Then comes the testing step: evaluating the output based on some criterion, typically regarding how well it matches some portion of the data. Our goal is to find those parameters that produce output that best meets our evaluation criterion.

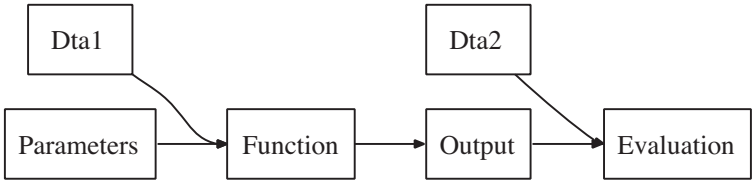


Figure 1.1 A flowchart for distribution fitting, linear regression, maximum likelihood methods, multilevel modeling, simulation (including agent-based modeling), data mining, non-parametric modeling, and various other methods. [Online source for the diagram: `models.dot`.]

The Ordinary Least Squares (OLS) model is a popular and familiar example, pictured in Figure 1.2. [If it is not familiar to you, we will cover it in Chapter 8.] Let $\mathbf{X}$ indicate the independent data, $\boldsymbol{\beta}$ the parameters, and $\mathbf{y}$ the dependent data. Then the function box consists of the simple equation $\mathbf{y}_{\text{out}} = \mathbf{X}\boldsymbol{\beta}$, and the evaluation step seeks to minimize squared error, $(\mathbf{y} - \mathbf{y}_{\text{out}})^2$.

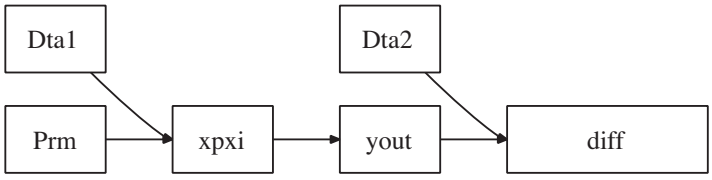


Figure 1.2 The OLS model: a special case of Figure 1.1.

For a simulation, the function box may be a complex flowchart in which variables are combined non-linearly with parameters, then feed back upon each other in unpredictable ways. The final step would evaluate how well the simulation output corresponds to the real-world phenomenon to be explained.

The key computational problem of statistical modeling is to find the parameters at

the beginning of the flowchart that will output the best evaluation at the end. That is, for a given function and evaluation in Figure 1.1, we seek a routine to take in data and produce the optimal parameters, as in Figure 1.3. In the OLS model above, there is a simple, one-equation solution to the problem: $\boldsymbol{\beta}_{\text{best}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$. But for more complex models, such as simulations or many multilevel models, we must strategically try different sets of parameters to hunt for the best ones.

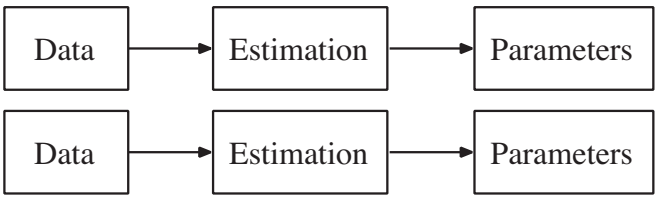


Figure 1.3 Top: the parameters which are the input for the model in Figure 1.1 are the output for the estimation routine.
Bottom: the estimation of the OLS model is a simple equation.

And that's the whole book: develop models whose parameters and tests may discover and verify interesting patterns in the data. But the setup is incredibly versatile, and with different function specifications, the setup takes many forms. Among a few minor asides, this book will cover the following topics, all of which are variants of Figure 1.1:

- Probability: how well-known distributions can be used to model data
- Projections: summarizing many-dimensional data in two or three dimensions
- Estimating linear models such as OLS
- Classical hypothesis testing: using the Central Limit Theorem (CLT) to ferret out apophenia
- Designing multilevel models, where one model's output is the input to a parent model
- Maximum likelihood estimation
- Hypothesis testing using likelihood ratio tests
- Monte Carlo methods for describing parameters
- "Nonparametric" modeling (which comfortably fits into the parametric form here), such as smoothing data distributions
- Bootstrapping to describe parameters and test hypotheses

THE SNOWFLAKE PROBLEM, OR A BRIEF HISTORY OF STATISTICAL COMPUTING

The simplest models in the above list have only one or two parameters, like a Binomial(n, p) distribution which is built from n identical draws, each of which is a success with probability p [see Chapter 7]. But draws in the real world are rarely identical—no two snowflakes are exactly alike. It would be nice if an outcome variable, like annual income, were determined entirely by one variable (like education), but we know that a few dozen more enter into the picture (like age, race, marital status, geographical location, et cetera).

The problem is to design a model that accommodates that sort of complexity, in a manner that allows us to actually compute results. Before computers were common, the best we could do was analysis of variance methods (ANOVA), which ascribed variation to a few potential causes [see Sections 7.1.3 and 9.4].

The first computational milestone, circa the early 1970s, arrived when civilian computers had the power to easily invert matrices, a process that is necessary for most linear models. The linear models such as ordinary least squares then became dominant [see Chapter 8].

The second milestone, circa the mid 1990s, arrived when desktop computing power was sufficient to easily gather enough local information to pin down the global optimum of a complex function—perhaps thousands or millions of evaluations of the function. The functions that these methods can handle are much more general than the linear models: you can now write and optimize models with millions of interacting agents or functions consisting of the sum of a thousand sub-distributions [see Chapter 10].

The ironic result of such computational power is that it allows us to return to the simple models like the Binomial distribution. But instead of specifying a fixed n and p for the entire population, every observation could take on a value of n that is a function of the individual's age, race, et cetera, and a value of p that is a different function of age, race, et cetera [see Section 8.4].

The models in Part II are listed more-or-less in order of complexity. The infinitely quotable Albert Einstein advised, "make everything as simple as possible, but not simpler." The Central Limit Theorem tells us that errors often are Normally distributed, and it is often the case that the dependent variable is basically a linear or log-linear function of several variables. If such descriptions do no violence to the reality from which the data were culled, then OLS is the method to use, and using more general techniques will not be any more persuasive. But if these assumptions do not apply, we no longer need to assume linearity to overcome the snowflake problem.

THE PIPELINE A statistical analysis is a guided series of transformations of the data from its raw form as originally written down to a simple summary regarding a question of interest.

The flow above, in the statistics textbook tradition, picked up halfway through the analysis: it assumes a data set that is in the correct form. But the full pipeline goes from the original messy data set to a final estimation of a statistical model. It is built from functions that each incrementally transform the data in some manner, like removing missing data, selecting a subset of the data, or summarizing it into a single statistic like a mean or variance.

Thus, you can think of this book as a catalog of pipe sections and filters, plus a discussion of how to fit pipe sections together to form a stream from raw data to final publishable output. As well as the pipe sections listed above, such as the ordinary least squares or maximum likelihood procedures, the book also covers several techniques for directly transforming data, computing statistics, and welding all these sections into a full program:

- Structuring programs using modular functions and the *stack* of *frames*
- Programming tools like the debugger and profiler
- Methods for reliability testing functions and making them more robust
- Databases, and how to get them to produce data in the format you need
- Talking to external programs, like graphics packages that will generate visualizations of your data
- Finding and using pre-existing functions to quickly estimate the parameters of a model from data.
- Optimization routines: how they work and how to use them
- Monte Carlo methods: getting a picture of a model via millions of random draws

To make things still more concrete, almost all of the sample code in this book is available from the book's Web site, linked from `http://press.princeton.edu/titles/8706.html`. This means that you can learn by running and modifying the examples, or you can cut, paste, and modify the sample code to get your own analyses running more quickly. The programs are listed and given a complete discussion on the pages of this book, so you can read it on the bus or at the beach, but you are very much encouraged to read through this book while sitting at your computer, where you can run the sample code, see what happens given different settings, and otherwise explore.

Figure 1.4 gives a typical pipeline from raw data to final paper. It works at a number of different *layers of abstraction*: some segments involve manipulating individual numbers, some segments take low-level numerical manipulation as given and op-

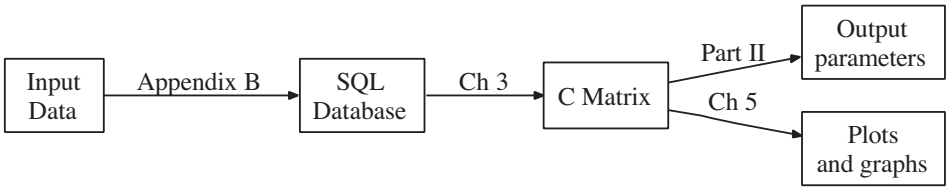


Figure 1.4 Filtering from input data to outputs. [Online source: `datafiltering.dot`]

erate on database tables or matrices, and some segments take matrix operations as given and run higher-level hypothesis tests.

The lowest level Chapter 2 presents a tutorial on the C programming language itself. The work here is at the lowest level of abstraction, covering nothing more difficult than adding columns of numbers. The chapter also discusses how C facilitates the development and use of *libraries*: sets of functions written by past programmers that provide the tools to do work at higher and higher levels of abstraction (and thus ignore details at lower levels).[2]

For a number of reasons to be discussed below, the book relies on the C programming language for most of the pipe-fitting, but if there is a certain section that you find useful (the appendices and the chapter on databases comes to mind) then there is nothing keeping you from welding that pipe section to others using another programming language or system.

Dealing with large data sets Computers today are able to crunch numbers a hundred times faster they did a decade ago—but the data sets they have to crunch are a thousand times larger. Geneticists routinely pull 550,000 genetic markers each from a hundred or a thousand patients. The US Census Bureau's 1% sample covers almost 3 million people. Thus, the next layer of abstraction provides specialized tools for dealing with data sets: databases and a query language for organizing data. Chapter 3 presents a new syntax for talking to a database, Structured Query Language (SQL). You will find that many types of data manipulation and filtering that are difficult in traditional languages or stats packages are trivial—even pleasant—via SQL.

[2] Why does the book omit a linear algebra tutorial but include an extensive C tutorial? Primarily because the use of linear algebra has not changed much this century, while the use of C has evolved as more libraries have become available. If you were writing C code in the early 1980s, you were using only the standard library and thus writing at a very low level. In the present day, the process of writing code is more about joining together libraries than writing from scratch. I felt that existing C tutorials and books focused too heavily on the process of writing from scratch, perpetuating the myth that C is appropriate only for low-level bit shifting. The discussion of C here introduces tools like package managers, the debugger, and the `make` utility as early as possible, so you can start calling existing libraries as quickly and easily as possible.

As Huber [2000, p 619] explains: "Large real-life problems always require a combination of database management and data analysis. . . . Neither database management systems nor traditional statistical packages are up to the task." The solution is to build a pipeline, as per Figure 1.4, that includes both database management and statistical analysis sections. Much of graceful data handling is in knowing where along the pipeline to place a filtering operation. The database is the appropriate place to filter out bad data, join together data from multiple sources, and aggregate data into group means and sums. C matrices are appropriate for filtering operations like those from earlier that took in data, applied a function like $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$, and then measured $(\mathbf{y}_{\text{out}} - \mathbf{y})^2$.

Because your data probably did not come pre-loaded into a database, Appendix B discusses text manipulation techniques, so when the database expects your data set to use commas but your data is separated by erratic tabs, you will be able to quickly surmount the problem and move on to analysis.

Computation The GNU Scientific Library works at the numerical computation layer of abstraction. It includes tools for all of the procedures commonly used in statistics, such as linear algebra operations, looking up the value of F, t, χ^2 distributions, and finding maxima of likelihood functions. Chapter 4 presents some basics for data-oriented use of the GSL.

The Apophenia library, primarily covered in Chapter 4, builds upon these other layers of abstraction to provide functions at the level of data analysis, model fitting, and hypothesis testing.

Pretty pictures Good pictures can be essential to good research. They often reveal patterns in data that look like mere static when that data is presented as a table of numbers, and are an effective means of communicating with peers and persuading grantmakers. Consistent with the rest of this book, Chapter 5 will cover the use of Gnuplot and Graphviz, two packages that are freely available for the computer you are using right now. Both are entirely automatable, so once you have a graph or plot you like, you can have your C programs autogenerate it or manipulate it in amusing ways, or can send your program to your colleague in Madras and he will have no problem reproducing and modifying your plots.[3] Once you have the basics down, animation and real-time graphics for simulations are easy.

[3]Following a suggestion by Thomson [2001], I have chosen the gender of representative agents in this book by flipping a coin.