Please see the online version of this article for supplementary materials.

Teacher's Corner

Computing in the Statistics Curricula

Deborah NOLAN and Duncan TEMPLE LANG

The nature of statistics is changing significantly with many opportunities to broaden the discipline and its impact on science and policy. To realize this potential, our curricula and educational culture must change. While there are opportunities for significant change in many dimensions, we focus more narrowly on computing and call for computing concepts to be integrated into the statistics curricula at all levels. Computational literacy and programming are as fundamental to statistical practice and research as mathematics. We advocate that our field needs to define statistical computing more broadly to include advancements in modern computing, beyond traditional numerical algorithms. Information technologies are increasingly important and should be added to the curriculum, as should the ability to reason about computational resources, work with large datasets, and perform computationally intensive tasks. We present an approach to teaching these topics in combination with scientific problems and modern statistical methods that focuses on ideas and skills for statistical inquiry and working with data. We outline the broad set of computational topics we might want students to encounter and offer ideas on how to teach them. We also discuss efforts to share pedagogical resources to help faculty teach this modern material (including supplemental materials).

KEY WORDS: Computational literacy; Curriculum reform; Information technology.

1. INTRODUCTION

The main message of this article is that the digital age is having a profound impact on statistics and the nature of data analysis, and these changes necessitate reevaluation of the training and education practices in statistics. In particular, computing is an increasingly important and necessary aspect of a statistician's work, and needs to be incorporated more fully into statistics training. Successful statisticians must be facile with the computer, for they are expected to be able to access data from various sources, apply the latest statistical methodologies, and communicate their findings to others in novel ways and via new media. In addition, researchers exploring new statistical methodology rely on computer experiments and simulation to explore the characteristics of methods as an aid to formalizing their mathematical framework. We believe that for the field of statistics to have its greatest impact on policy and science, statisticians must seriously reflect on these major changes and their implications for statistics education.

The ability to express statistical computations is an essential skill (analogous to algebra and analysis/calculus) for practicing data analysts, statistical researchers, and students. But, how well are we, as a community of statistics educators, preparing our students for this modern era of statistics research and practice? Peck and Chance (2007) hailed the need for educators to try to find out what students are actually learning as they complete the course work for the statistics major. They suggested beginning with an "open and frank discussion of the following question: When your students graduate, what is something observable that you think they ought to be able to do?" We follow their lead and ask the questions: When they graduate, what ought our students be able to do computationally, and are we preparing them adequately in this regard? Do we provide students the essential skills needed to engage in statistical problem solving and keep abreast of new technologies as they evolve? Do our students build the confidence needed to overcome computational challenges to, for example, reliably design and run a synthetic experiment or carry out a comprehensive data analysis? Overall, are we doing a good job preparing students who are ready to engage in and succeed at statistical inquiry? We believe that in general, we fall short of these goals, and call on the statistics community to work together, to challenge ourselves and each other, and to make a significant cultural shift to embrace computing and integrate it fully into statistics undergraduate major and graduate programs.

The recent article, "What Is Statistics?" (Brown and Kass 2009), called attention to the changing landscape for the field of statistics where success requires "highly flexible problem-solving strategies," yet the authors observed a "worrisome ten-

Deborah Nolan is Statistician, Department of Statistics, University of California, 367 Evans Hall MC 3860, Berkeley, CA 94720-3860 (E-mail: *nolan@stat.berkeley.edu*). Duncan Temple Lang is Statistician, Department of Statistics, University of California, 4210 Mathematical Sciences Building, One Shield Avenue, Davis, CA 95616 (E-mail: *duncan@wald.ucdavis.edu*). The authors gratefully acknowledge support from the National Science Foundation grants DUE 0618865 and DMS 0636667. The authors thank the reviewers and John Monahan for their helpful comments, which led to an improved version of this article.

 DOI: 10.1198/tast.2010.09132

dency" to "attack problems using blunt instruments and naive attitudes." Brown and Kass attributed part of the problem to our courses and degree programs. They pointed out that historically "mathematical thinking influenced both research and infrastructure" in statistics departments and currently may not be serving the field well. It is really important that statisticians are beginning to reflect on the changes in the field and how these impact training programs. We share many of the same concerns and solutions as described by Brown and Kass (2009), and call for the statistics community to examine the legacy of computational training for statisticians. The skill set needed by a statistician even 20 years ago is very different from what is needed today.

1.1 Three Key Components

Computational ability supports statistical inquiry and is vital to all facets of a statistician's work. Yet, it occupies an astonishingly small proportion of the statistics curricula. Many statisticians would agree that there should be more computing in the statistics curriculum and that statistics students need to be more computationally capable and literate. However, it can be difficult to determine what specifically to teach and how to practically change the curriculum. In our experience with designing and teaching new courses in computing, the following considerations have come to the forefront.

1. *Broaden statistical computing.* Within the past five to ten years, an ever larger array of computing topics have impacted the work of practicing statisticians and this is only likely to continue to increase as the Web is used to disseminate data in rich new ways. While the traditional area of computing related to the study of numerical analysis and algorithms for statistical methods (often referred to as "computational statistics") is still very important to the field, other topics have also become significant, and to different categories of students. Today, and in the future, statisticians must access and integrate large amounts of data via Web services and databases, manipulate complex data (e.g., text, network graphs) into forms more conducive to statistical analysis, and produce interesting statistical presentations of data as exemplified by GapMinder (GapMinder Foundation 2008). The importance of these innovations on the access, analysis, and presentation of data make a strong case for broadening the statistics curriculum to include these nontraditional topics.

2. *Deepen computational reasoning and literacy.* In many regards, computing skills are similar to mathematical skills, which are important to statistics not in their own right but in how they allow us to reason about and express statistical ideas and techniques. Statistics students, at the major and graduate levels, must be able to express themselves through computations, understand the fundamental concepts common to programming languages, and discuss and reason about computational problems precisely and clearly. Instruction must move beyond how-to examples, idioms, and templates for students to copy. Students need to gain explicit experience with programming an algorithm and with programming language concepts such as variables, assignments, flow control, functions, parameters, data structures, input and output, error handling, debugging, and so on. Whether they enter the workforce from an undergraduate degree in statistics or enter graduate programs and research careers, future statisticians will encounter an ever changing array of "current" technologies, data formats, and programming languages. They need a solid understanding of programming concepts, paradigms, and fundamentals to meet these emerging changes and challenges and to solve novel computing problems.

3. *Compute with data in the practice of statistics.* We believe that statistical computing, like statistical methods, should be taught in the context of statistical practice to give students both the motivation to interact with data and the experience needed to be successful in their future statistical endeavors. The nature of "computing with data" needs to be addressed by working on real computational problems that arise from data acquisition, statistical analysis, and reporting. One important side effect of this approach is that statistical computing could be taught in a way that offers an alternative approach to teaching statistical concepts (both elementary and advanced). Computing offers a possible approach to addressing the larger issues facing our entire curriculum: how to teach statistical thinking and problem solving in the scientific context. Furthermore, computing can be used to teach statistical methodology in a quite complementary and different manner than mathematics, where students actively "construct," that is, program, and explore methods.

These three key aspects (broaden statistical computing to include emerging areas, deepen computational reasoning skills, and combine computational topics with data analysis in the practice of statistics) are discussed in greater detail in Sections 2, 3, and 4, respectively, of this article. The question of what specifically should be taught is addressed in Section 5. There we present a myriad of possible topics, which we collect into six main groups: (1) Fundamentals in statistical computing with data, (2) Information technologies, (3) Computational statistics (e.g., numerical algorithms) for implementing statistical methods, (4) Advanced statistical computing, (5) Data visualization, and (6) Integrated development environments. We also provide examples, sketches, and snippets of possible approaches to teaching these topics in Section 5 and as supplementary material to this article. Section 6 makes a final plea to the statistics community to take the challenge and work ambitiously to incorporate statistical computing into the statistics curricula and suggests some practical approaches that may help make such changes feasible.

In short, a good foundation and skills in computing are essential for all who participate in research and scientific inquiry, and increasingly so for statisticians. Specifically, we advocate that computing must be central to the statistics curriculum at the undergraduate and graduate levels. For example, at Berkeley, there are now three upper-division courses, one each in probability, statistics, and computing with data, that form the core of the major program. Of course, not all institutions have the flexibility to develop a statistics course on computing with data, but it is crucial to have a basic course that addresses computational reasoning and the fundamentals of programming with data, which is very different from a typical introductory programming class. Such a course may be taught as part of an applied mathematics or computer science major; for example, Macalester College's mathematics department offers the MATLAB-based course, Scientific Programming, and uses the text by Kaplan

(2004). In addition to a core computational course, we strongly advocate integrating computational problems and vocabulary into traditional statistics courses.

1.2 Our Backgrounds

We have been thinking about and working on making changes in these directions for several years. Our perspectives come from our different backgrounds and experiences. Temple Lang spent many years in Bell Labs in industrial research and development, and Nolan received formal training in programming while working for IBM 30 years ago. A significant aspect of Temple Lang's research focuses on both developing statistical computing environments and integrating information technologies with statistics. Nolan's general work in pedagogy, and particularly in the use of technology in statistics education, provides yet another viewpoint. Both of us teach at the University of California, on the Berkeley (Nolan) and Davis (Temple Lang) campuses. For the past six years, we have been involved in curriculum reform at our respective campuses. Temple Lang is in the process of introducing a new sequence of courses and topics in statistical computing in the graduate and undergraduate programs at Davis, and Nolan and Temple Lang have designed new undergraduate courses in computing and data technologies that are core components of their major programs.

We have developed these undergraduate courses in close cooperation, yet they differ in that they focus on different levels of students: at UC Davis the audience has been a mixture of graduate and undergraduate students, and at UC Berkeley the students are undergraduates in statistics, applied math, and other science majors. The courses are also of different lengths because Berkeley has a 15-week semester and Davis a 10-week quarter. Both center on programming fundamentals and thinking and information technologies and how they connect to statistics. They also incorporate problems with real, complex data arising in industry and research settings, and they engage students in statistical practice, exposing them to some modern/nontraditional statistical methodology.

Most recently our perspectives have also been shaped by our shared experience organizing a series of NSF-funded workshops to assist faculty in acquiring the knowledge, skills, and teaching practices in new areas of statistical computing. The first workshop, held in 2007, brought together computing specialists and industrial research consultants to help determine modern syllabi and curricula involving computing. Subsequent workshops focused on providing instructors with the background and skills needed to teach statistical computing courses, and collectively developing examples or case studies of modern data-analysis projects to share with the statistics community. To support continued discussion and assistance for faculty and to build a community of educators interested in incorporating computing into the statistics curriculum and sharing course materials, we have also created electronic mailing lists, discussion boards, and a wiki. See the Web page by Nolan, Temple Lang, and Hansen (2007a) for a more complete description of these efforts and materials, and see the supplemental material to this article for example datasets and simulation studies.

2. BROADEN STATISTICAL COMPUTING

In answer to the question, "What do you regard as the greatest contribution to statistics over the last 40 years?" Hartigan (Barry 2005) chose the S language (Chambers and Wilks 1988) and the Internet:

> the Internet has made a great contribution in that data that used to be hard to get hold of are much more available these days . . . people are much more realistic when thinking statistically . . . They pay attention to data much more . . . These [the S language and the Internet] are both statistical computing things that have had a very big effect on statistics and will continue to have an effect.

We wholeheartedly agree. Traditionally, the computing taught in the statistics curriculum falls into one of two camps, (a) numerical analysis and algorithms for statistical methods; and (b) the nuts and bolts of how to use a programming language. We think that statistical computing encompasses more than these topics. Statisticians at all levels increasingly deal with large amounts of data from many and varied sources (increasingly via the Web) and the challenges to data analysis start well before the computational steps involved in model fitting. For these reasons, many emerging topics in information technologies are becoming important to statistics. Additionally, various aspects of high-performance computing are becoming important and will change our understanding of algorithms, which are still relevant to the field.

For many, merely defining the terms Web, data, and information technologies may prove difficult and the separation between programming and data technologies is not black and white. We think of data technologies as computational tools, techniques, and paradigms that allow access to and transformation of data from varied sources and formats and also the presentation of information, results, and conclusions in rich, dynamic ways. These technologies include: shell commands or Perl/Python programs to preprocess data; regular expressions for text manipulation; relational database management systems for storing and accessing data; Web services for structured access to remote data and services; the eXtensible Markup Language (XML) used for many purposes, including exchanging self-describing data and supporting Web services; and Keyhole Markup Language (KML), JavaScript, and Flash to publish Web displays or "mash-ups" of data that are interactive. That is, information technologies are essential tools by which researchers access and present data and results. As the Web continues to grow and interesting data are made available in rich ways on the Web, these technologies are becoming increasingly important for a statistician's work.

In the spirit of Peck and Chance, we ask, What do our students do when they graduate? We have found that Bachelors and Masters students who enter the workforce spend much of their efforts retrieving, filtering, and cleaning data and doing initial exploratory data analysis. These responsibilities increasingly demand working with different data technologies and having general programming skills. The potential for interesting and rich interactions for a statistician is greatly increased if he or she has a good knowledge of information technologies. One former student recently wrote to us about exactly that:

> I am currently working at a consulting firm that specializes in statistical and economic research and data analysis for large cor-

> porations. . . . Every day I work with data, and whether it is running regressions, cleaning data, finding summary statistics, parsing documents, or working in different database environments, [this statistical computing class] gave me the tools and foundation to succeed in my current position and gave me the confidence to land the job in the first place.

Computing with data is an issue in which our field should be taking the lead in order to try to improve the level of data competency and literacy within the broader scientific community. Also, if "companies" are not engaged in these activities now, they soon will be and we as educators should enable them to do so by producing graduates who can do new things and avail themselves of the massive growth in available data and sources. In other words, we should not simply be teaching students skills that industry currently values in order to get a job. We should also be developing our students into discerning, critical-thinking active participants of an emerging data-driven society. Academia has an opportunity to lead and aid in the new era of ubiquitous data and information.

The goal of teaching computing and information technologies is to remove obstacles to engagement with a problem. If students gain a basic familiarity with various technologies, they can carry these skills with them and build on them to address new problems. In this way, they are more autonomous, more "can do," and less dependent on others to define the problem and present them with the data. Technology will continually evolve, which is why it is crucial to teach students the art of learning new technologies so that they can understand, evaluate, and compare them as they emerge. More important than the details of the specific technologies, we should teach students to learn *how to learn* about new technologies on their own, for example, to cull information from on-line documentation, tutorials, and resources and to identify important concepts. We should not expect them to learn this skill on their own, however. We must teach this important material and this requires solid foundations in computing principles and reasoning.

3. DEEPEN COMPUTATIONAL REASONING AND LITERACY

The ability to express computations is an essential skill for practicing data analysts, statistical researchers, and students that improves every aspect of their work. Furthermore, as analysis and research continue to become more complex, more novel computational approaches are required and a deeper understanding of computational technologies can make intractable problems feasible. The ability to reason about and express the computations is not the end-goal, but it is a vital aspect of the overall task and one that is involved in and supports all facets of a statistician's work. Friedman (2001) noted almost 10 years ago that:

> Computing has been one of the most glaring omissions in the set of tools that have so far defined Statistics. Had we incorporated computing methodology from its inception as a fundamental statistical tool (as opposed to simply a convenient way to apply our existing tools) many of the other data related fields would not have needed to exist. They would have been part of our field.

We agree with Friedman and fear that if we do not address this challenge more coherently, statistics will become marginalized and less relevant at a time when its importance is growing dramatically. However, we do offer an opinion as to how to achieve this goal of adding computing to the statistician's toolbox. Foremost, to gain skills and competency in computational reasoning, "programming" courses must cover broad concepts as well as the specifics and recipes. While statistics students need to learn practical details of programming (e.g., language syntax), faculty must strive to teach higher level concepts, including a computing vocabulary and computational thinking that will enable students to reason about and discuss computational problems precisely and clearly. As computing and information technologies continue to evolve rapidly, it is essential that students develop a good foundation rather than a thin memorization of specifics so that they are able to reason about computational tasks and continue to learn new aspects of computation. Additionally, a lack of computational reasoning skills makes it difficult for statisticians to work in a team where others are computationally capable, independent, and autonomous. We believe that adding some structure and guidance in teaching computing fundamentals yields large professional gains for students, research assistants, and professionals and our field generally.

Many statisticians advocate—or at least practice—the approach in which students are told to learn how to program by themselves, from each other, or from their teaching assistant in a two-week "crash course" in basic syntax at the start of a course. Let us reflect on how effective this approach has been. Can our students compute confidently, reliably, and efficiently? We find that this do-it-yourself 'lite' approach sends a strong signal that the material is not of intellectual importance relative to the material covered in lectures. In addition, students pick up bad habits, misunderstandings, and, more importantly, the wrong concepts. They learn just enough to get what they need done, but they do not learn the simple ways to do things nor take the time to abstract what they have learned and assimilate these generalities. Their initial knowledge shapes the way they think in the future and typically severely limits them, making some tasks impossible.

We believe that a main goal in teaching programming is for students to be able to extract concepts from a specific problem and transfer concepts to the specifics of other languages and environments. Taught in isolation, programming languages are idiosyncratic and arcane, but when taught more generally, the commonality and patterns emerge and provide a significantly simpler viewpoint and much more useful, general skills that will serve them well in the future. We advocate that students learn a complete, general-purpose programming language such as R (R Development Core Team 2006) or MATLAB (Mathworks 2009) where they can create new algorithms and functionality and express statistical ideas and computations at a relatively high level. We could even use languages such as Python or Perl, but these are less statistically focused and would involve more start-up costs within a statistics class. Some students, such as graduate students who are researching new methods, will need to learn lower level languages (e.g., C or FORTRAN), but most will be well equipped with a foundation based on languages such as MATLAB and R.

4. COMPUTING WITH DATA IN THE PRACTICE OF STATISTICS

Why teach statistical computing in the context of solving scientific problems through data analysis? We offer several reasons.

- Programming is technical and often frustrating, but when embedded in exploring data, drawing plots, looking for anomalies, making conjectures, and looking for supporting evidence, the students learn the computational aspects as part of an interesting, challenging, exciting, confidence-building process. The computer provides feedback to the students and helps to guide their activities and learning.
- For most students, it is a big leap from practicing basic programming skills to embracing problem-solving methodologies and general computing principles. Students, ideally, gain this experience as they behave like scientists/statisticians who work with data, that is, when they compute with data in the context of solving a scientific problem.
- With this approach, students are exposed to a much more subjective, creative activity than typically encountered in traditional computing and statistics methodology classes, and as a result gain a deeper, richer appreciation for the practice of statistics.
- Moreover, this approach typically involves multiple computational aspects, for example, text manipulation with regular expressions, a simulation study, and advanced graphics, and in this way, students learn to be facile with computational tools to express ideas and map concepts to programming instructions.
- Statistical computing topics that are integrated with data and context add a new pedagogical dimension to the entire statistics curriculum and can expose students to methodology that they would not typically encounter.

It is commonly accepted that statistics should be taught in context. Many statistics educators have argued that teaching "mathematical statistics" must include experience with data and real problems, and there is evidence that courses and curricula are changing in this direction. See the works of Nolan and Speed (2000), Ramsey and Schafer (2002), Utts and Heckard (2003). We believe the same approach is needed for teaching statistical computing. In this case, however, these changes require rebuilding much of the usual infrastructure for teaching in order to integrate support in the classroom for statistical *and* computational reasoning.

In the courses we have designed, we teach computational topics through a combination of lectures, computer labs, hands-on computing tasks in homework, and group projects. The projects are key because they integrate multiple computational topics in the context of a modern data problem. Students gain hands-on experience with statistical concepts flowing from contextual problem solving with data, and they make their own discoveries by posing and answering questions rather than solely fitting models or using "this week's lecture's methodology" as a computing exercise. These projects form an important part of the students' work. They get the experience of working on a statistical problem from beginning to end, gathering data, doing the analysis, and presenting the results. We attempt to foster a culture of active engagement, where students work with open-ended questions, are continually exposed to modern methods and basic statistical concepts, and encouraged to be creative throughout several stages of data analysis. The aim is for students to learn through 'effortful study' as Weiman (2008) described:

> True understanding only comes through the student actively constructing their own understanding through a process of mentally building on their prior thinking and knowledge through *effortful* study.

Creating class projects requires finding a substantial and interesting dataset with an associated scientific or social problem and then designing a sequence of feasible tasks that lead to the pedagogical goal. In our experience, this typically means trying four or more datasets to find one that fits all the necessary criteria. [See the supplemental material to this article for a list of datasets and simulation studies that we have found effective as class projects, and also the Data Expo competition organized by the ASA Statistical Computing and Graphics Sections (Wickham 2009b).] It is a lot of work, and we strongly advocate pooling resources so that the materials needed to teach the topics are available in formats that can be quickly adapted and customized for different situations. To this end, in 2009 we began working with a group of faculty to create, gather, and disseminate materials that can be used for projects, assignments, and classroom materials (Nolan, Temple Lang, and Hansen 2007a). With these efforts, we hope to seed the statistical community with resources for introducing and teaching computing at the major and graduate levels, much as CAUSEweb (2009) and DASL Project (2009) do for introductory statistics courses.

5. TOPICS IN COMPUTING FOR DATA

What specific topics should be taught, and how should they be organized? Figure 1 displays a plethora of topics that fall under the statistical computing rubric. Not all students need exposure to all topics or to particular topics at the same level. We have collected these topics into six groups: (1) Fundamentals in statistical computing with data, (2) Information technologies and Web technologies, (3) Computational statistics (numerical algorithms), (4) Advanced statistical computing, (5) Data visualization, and (6) Integrated development environments. We note that many topics fall within multiple groups, and this division is not cut-and-dried, nor is the set of topics comprehensive. This figure is a starting point for developing a logical taxonomy for computing in statistics. We have placed the figure on the Web *http://www.stat.berkeley.edu/users/statcur/* in order to annotate it and adapt it as topics are further clarified and formalized into a taxonomy.

Although one might consider teaching a separate course on each group of topics, we do not suggest that as the only, or even the ideal, approach. Topics within each of these categories can be mixed and matched across categories to produce courses for different levels and different needs of students, and they can be incorporated into traditional statistics courses. For undergraduates, we strongly endorse a course in programming and statistical computing with a heavy mix of exploratory data analysis

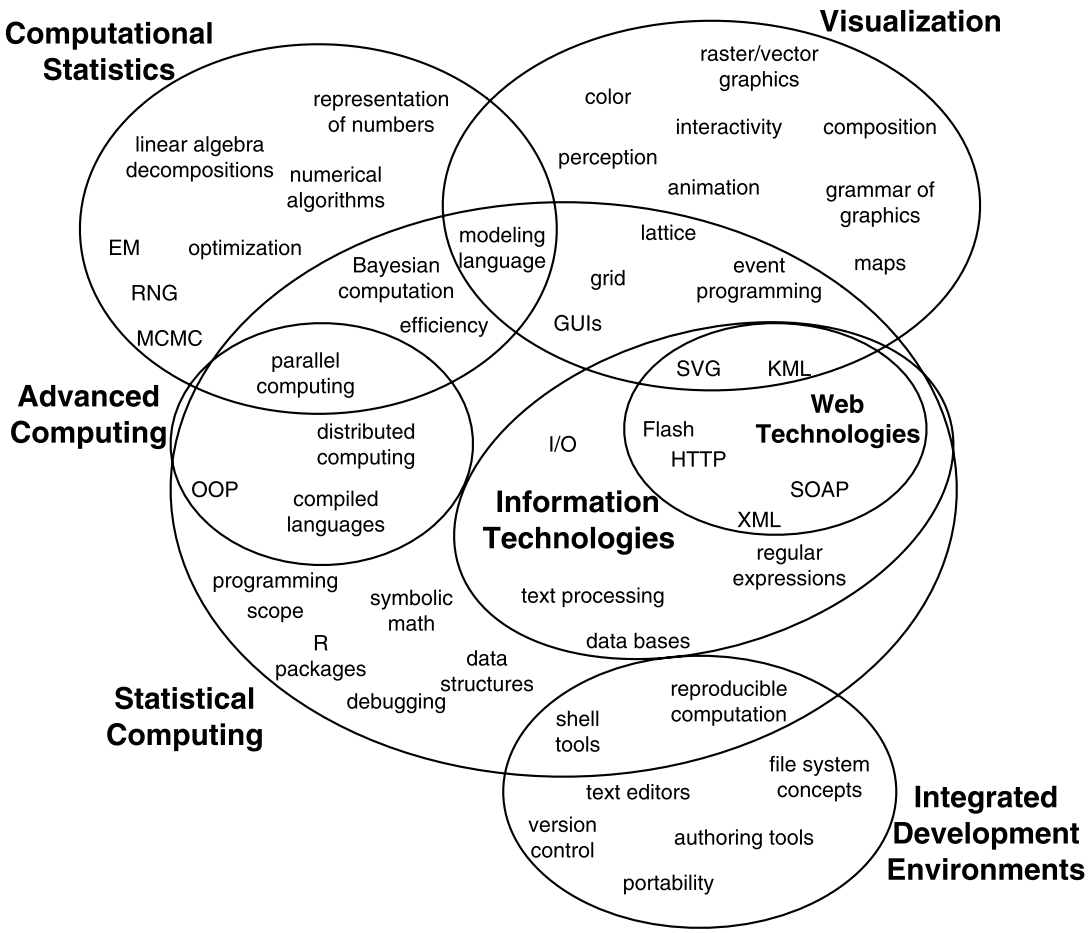

Figure 1. This image shows the broad array of computational topics relevant to statistics. These topics are loosely arranged into several groups. As seen here, many topics fall into multiple categories, and given the two-dimensional constraints, it is difficult to show all of the overlap. The list is not meant to be exhaustive, but rather indicative of the types of topics that we consider relevant to statistical practice and research. It shows additional, nontraditional statistical computing topics that are of increasing importance in this changing world of data sciences.

and modern statistical methods. Table 1 provides a list of topics for an undergraduate course that we teach that covers statistical computing, information technology, visualization, and modern data analysis. In general, at the undergraduate level, we emphasize basic proficiency, problem solving, and familiarity with useful programming environments. We recommend that a

Table 1. Provided here is a list of possible topics for a 15-week undergraduate course that covers topics in programming in R, information technologies, and visualization. The column on the left provides the general topic and the column on the right provides one or two specific examples for the topic. For examples of how this material might be taught, including ideas for projects, see Sections 5.1 and 5.2.

General topic	Examples
Introduction to R environment	Language syntax
Visualization and graphics	Simple graphics, vocabulary, composition
Fundamental data structures	Vectors, lists & data frames, factors, subsetting
Digital information	File system and formats
Programming	Control flow, basics of writing functions
UNIX shell	Batch programming, basic stream tools
Modern methods	Recursive partitioning, naive Bayes
Text processing	Regular expressions (in R)
Advanced graphics	Grid, Google Earth, Scalable Vector Graphics (SVG)
Writing functions	Breaking tasks into separate functions, nested call frames, recursion
Simulation	Stochastic processes, random number generation
Debugging, optimization, and efficiency	Stepping through code, exceptions, profiling, efficient idioms
Relational databases and SQL	The SELECT statement
HTML and structured content	Report generation
HTML/XML programming	Web scraping

course with this focus be a core requirement for all undergraduate majors.

For graduate students, we propose that they have a course in the fundamentals of programming and statistical computing with data. In addition, depending on their area of research, we recommend extensive exposure to information technologies, computational statistics, and advanced computing. These additional topics might be taught through regularly offered computing courses and/or woven into traditional statistics courses. One example of an advanced computing class in high-performance computing is provided in Section 5.4.

In the following subsections, we discuss each of these groups of topics. We highlight a few different important aspects in teaching each particular group, including how one might go about teaching this material and why. Several example syllabi and course materials have been collected on the Web and Wiki (Nolan, Temple Lang, and Hansen 2007a, 2007b), if the reader wishes to see more examples and more details.

5.1 Fundamentals in Statistical Computing

We earlier made the case that both undergraduate majors and graduate students need to have facility in computational reasoning and thinking. How might one go about teaching these fundamentals? Our aim is to first teach students how to think about a language such as R or MATLAB, and use it to express concepts. That is, we demonstrate the usage paradigm of the language, and from there, we move on to discuss programming concepts and computational techniques. In this programming environment, we find that students quickly gain experience with the syntax of simple assignment statements, computing with variables, and calling standard functions with input arguments. The students independently compose queries of data, and as they explore the programming environment, they receive instant feedback (and gratification).

Next we introduce data structures, and discuss why they came to be in the language and not other, alternative ones. Reading different types of data from files or over the Web and illustrating the different data types that result is an easy starting point. As we discuss representations of data, the students embark on explorations that require them to operate on data structures. We have found that exploratory data analysis and graphics provides a rich context in which to cover the essentials. This approach puts students in an active role, trying different things, figuring out what to try next, and thinking about why one command worked and a previous one did not. This retrospective analysis is essential for learning computing effectively and richly.

One example that we have had success with involves the signal strength data described in the next section. These data arrive as a ragged array where each row provides the signal strength emitted from a device at a particular location in a building and measured at a variable number of access points. Once the maximum number of access points is determined, the data can be organized with a column of signal strength measurements for each access point, with missing values for those locations where the access points detected no signal. Alternatively, the data can be organized into a "stacked" structure with one column for the signal strength measurement and another to identify the access point from which the measurement was taken. Given the size of the data, students must face issues of efficiency when creating the data structure, redundancy of information in the data structure, and ease of performing subset and plotting operations.

We have also found that simulation studies make for excellent projects and exercises in function writing. One such project is a simulation study of the Biham–Middleton–Levine Traffic Model (BML; Angel, Holroyd, and Martin 2005), which is a simple stochastic process that exhibits a phase transition. There are two types of cars moving on a grid; one type moves south to north and the other west to east. Cars populate the grid at random, and the occurrence of the phase transition relates to the density of cars. The algorithm for simulating this process depends on the representation of the information at a particular time, for example, as a matrix with values indicating whether the position on the grid is occupied by a west–east car, a south–north car, or unoccupied, or as a data frame where each row corresponds to a car in the grid and the columns provide information about its location and direction.

In addition, we believe it is important for students to be exposed to more than one language. The comparison of a second language helps solidify the similarities of the computational models of both languages as well as fundamental programming concepts. It also illustrates that different languages have different purposes. We have found that exposure to one or more of shell tools, regular expressions, SQL, and XPath offers a variety of perspectives, different evaluation models, and semantics. As an example, one assignment we give has students explore baseball data that are in a relational database (Lahman 2008). The *RSQLite* (James 2009) package allows them to issue SQL statements from within R to extract data from the database. An important aspect of this assignment is to determine where to perform the computations, that is, in the database before the data are brought into R, or in R. Thus students have the opportunity to directly compare the different evaluation models and semantics of the two languages. These languages are sufficiently different in nature and used for very different tasks so that discussing them does not confuse the students.

In the early stages of the course, we give students guidance on how to structure the composition of a function to make it easier to understand and be reused in other contexts and problems by others and by themselves. We describe language details such as default values for parameters, but we also discuss general principles of computing and program design. Again, this abstraction is vital if the students are to come away with general, reusable concepts and the ability to think about computational problems rather than just mimic.

Students solidify programming fundamentals and learn to discuss and reason about computational problems through work on longer projects. Early on, we often break up projects into parts with the first few tasks given as homework assignments. With this approach we: (i) model the practice of function writing, identifying subtasks and writing "helper" functions to handle these tasks; (ii) focus on one aspect of programming, such as how to debug and profile code; and (iii) demonstrate how to validate code by using diagnostic statistics on the output. Additionally, from a course management perspective, this approach of breaking a project into stages allows *us* to: help students subsequently manage a large project with greater success; discuss

solutions to intermediate problems in class; and provide alternative versions of functions that students can employ in the final project.

5.2 Information Technologies

When we teach information technologies, we introduce many of the important and emerging tools in the context of their use in statistical practice and research. Students learn to access complex data in "raw" forms such as documents, citations, log files, DNA sequences, or annotations. We discuss software development for the data analysis process, including text processing, accessing and creating relational databases, and working with web servers. While students may be able to take courses in computer science on these various topics, such courses may not be the best option. For example, with databases, statisticians are generally consumers and need not initially focus on the details of optimizations performed in the execution of a relational database query. Instead, they should understand the general principles of the relational database model and the common elements of the Structured Query Language (SQL) used to extract data from a database. This essential material can be covered in a few lectures, and can be combined with lectures on an array of other data technologies that are geared to the needs of modern data analysis.

The main topics included under information technologies were discussed in Section 2 (and displayed in Figure 1). These topics demand an example-driven approach to teaching in part because there are few teaching resources available. We describe three undergraduate projects that we have our students work on. They mix fundamentals in programming with information technologies, and help students develop their computational reasoning skills through putting the material learned in the classroom into practice with real, complex problems. We have developed this material into a modern, vibrant course that includes advances in statistical methodology and applications with data, which we describe in an upcoming book (Nolan and Temple Lang 2011).

One project involves a problem in geo-location using wireless signals. Students begin by familiarizing themselves with the data (King et al. 2008), which consist of signal strengths for multiple access points measured on a hand-held client device at fixed locations and orientations throughout a building. They find ways to reorganize the data into a form that is more amenable to further analysis. They explore and create interesting visualizations of these data, by, for example, overlaying contour plots of signal strength on a floor plan of the building or making three-dimensional plots and heat maps of signal strength. Students write functions to compute nearest-neighbor distances, and use this intuitive method (along with cross-validation) that is not traditionally taught to build and test a model for estimating the physical location of an object from the received signal strength at the access points.

A second project involves filtering spam email. The data consist of a collection of over 9000 emails (SpamAssassin 2007), provided as raw text files that have been classified and named according to whether they are spam or ham (regular email). Students use connections, string manipulation, and regular expressions to break the text files apart, identifying the header information, body, and attachments. Then groups of students write functions to extract information into dozens of variables, for example, the percent of yelling (capitalization) in the subject line, the number of attachments, presence of HTML, etc. Via exploratory data analysis they identify the derived variables that may be good predictors for spam. At this point they fit statistical models to predict spam using recursive partitioning/classification trees, and they use cross-validation to select tuning parameters such as choice of metric. In the final stages of the project, they assess, through visualization, how well their method works on test data that were set aside and explore patterns in the misclassifications.

Another project has students study the migratory patterns of an elephant seal that has been followed with a tracking device (Brillinger and Stewart 1998). Students fit smooth curves to the seal's migratory path; conduct a simulation study to compare the seal's path to a random walk on a great circle on the earth; and present their findings via an animated "mash-up" on Google Earth.

Through these projects, we attempt to instill in students a problem-solving ability so they will have a basic familiarity with computing technologies which they can carry with them in addressing new problems. This strategy also helps prepare students to meet the challenge of constantly evolving technologies because in the projects they learn the art of learning new technologies and are better able to understand, evaluate, and compare them as they emerge in the information world. Furthermore, as instructors in today's Web-based world of information exchange, we no longer need to think in units of textbooks but rather smaller units that can be combined creatively into courses, for example, well-documented case studies with data that cover multiple computational and methodological topics and have open-ended extensions.

5.3 Computational Statistics

By computational statistics, we mean the more traditional topics in numerical analysis and methods. Which specific topics should be taught has been addressed in detail by others (Gentle 2004; Lange 2004; Monahan 2004). These include linear algebra decompositions, numerical optimization, and aspects of approximation and numerical analysis. These are methods for obtaining solutions efficiently or approximately, or both, and often become necessary in advanced research. These classical topics are of importance and, all else being equal, students should master them. However, we wish to provoke thought about their importance relative to other potential topics in computing.

We think it is important to distinguish between two goals of teaching these topics: (i) to understand the components and underpinnings of existing software when choosing what software to use; and (ii) to create one's own software. For the first goal, students need to be aware of the computational issues and the circumstances under which an algorithm is best or even suitable. We suggest that this understanding not be limited to a computational course. Instead, these topics would be most effectively taught when *woven* into other more traditional statistical theory or methodology courses in which the need for efficient, stable algorithms is encountered.

For the second goal, we expect graduate students to be able to contribute robust and accurate software to support their research and to critically understand the techniques used in existing software. We advocate that the material be taught with extensive computer experiments to fully understand the characteristics of the algorithm and simultaneously teach aspects of experimental design for computer experiments. Therefore topics in numerical algorithms for statistical methods—essentially how to do statistical computations properly—presuppose adequate programming skills and are more appropriate as a follow-up to an introductory course in statistical programming and computing environments. Additionally, issues about software development are important and we discuss them in the next section.

5.4 Advanced Computing

In addition to programming languages, graduate students will often need to learn new paradigms, such as compiled languages and parallel/distributed computing. The latter is no longer an exotic, specialized topic but an increasingly commonly used technique for implementing scientific computations. Similarly, as statisticians continue to publish software implementing their methodological research, graduate students need to understand some essential principles of software engineering. Issues of portability, memory management, data structures, object-oriented programming, compilation, efficiency, profiling, unit testing, extensible design, leveraging existing software, and so on are very important in developing software for others to use. What might have been considered rare in computing a decade ago is becoming more important for doctoral students so that they can successfully function in, and contribute to, the scientific community and disseminate their scholarship in rich, new ways.

One version of an advanced course might focus on high-performance computing, that is, dealing with large datasets and computationally intensive methods. Such a course might provide a mix of formalism, infrastructure, tools, and applications that deal with efficient computing related to data analysis, simulation, scientific computing, and visualization. It would be a mix of how-tos and tutorials, surveying the topics, extracting the high-level, abstract concepts, and providing good techniques and practical approaches to problems. The aim would be to provide students with an understanding of the issues in the area of resource-constrained computing problems and to address practical solutions and understand their characteristics. It would focus on using high-level languages and the associated tools while mixing these with other facilities and languages to perform computations.

How might one organize such a course? One approach would be to begin with examples of computations that consume resources and are infeasible. This is done in R and MATLAB with comparisons of the two. Having addressed approaches for efficient computation in high-level interpreted languages, we would then examine lower-level, high-performance languages. Topics covered might be:

- General and simple approaches to reduce consumption of resources, such as constant-folding or invariant extraction from loops, pre-allocation of space, and elementary rewriting of code.
- Algorithms, profiling, and measurement of code, and reorganizing computations to speed them up or simply make them feasible.
- Estimation of resources that are needed for a particular task (i.e., memory and time/cycles) using both theoretical computations and experimental/empirical approaches including simulations and profiling.
- Algorithm-specific approaches to improve efficiency, such as out-of-memory or block algorithms, which illustrate the general technique of reorganizing the computations, and data reduction techniques.
- Integration of lower-level, high-performance languages, for example, C/C++, FORTRAN, and Java, with R and MATLAB, including writing interface code, compiling, linking and loading the native code, and debugging.
- Distributed and parallel computing, including the essential framework, the problem of synchronization, and approaches used to resolve this problem, and the effect of data distribution and aspects of random number generation.

5.5 Data Visualization

We include topics in visualization under the statistical computing umbrella because of their tight coupling with computing, and we put these topics into their own group because of the general importance of visualization. Furthermore, we have found that data visualization tends to be under-represented within the statistics curricula.

A goal of teaching data visualization is to provide a framework with which students can critique, compose, and create graphics that usefully display information from data, including the standard types of displays used in statistics. The on-line data visualization tools for shared visualization, for example, Many Eyes (IBM 2009) and Swivel (Dimov and Mulloy 2009), offer an abundance of graphics with accompanying data for students to deconstruct, critique, and reconstruct in, for example, R. In addition to teaching the mechanics of creating graphics, other topics include cognitive perception and aspects such as shape, size, texture, and color models. This material is quite different from the field of scientific visualization which tends to focus on computer modeling of objects, surfaces, and shapes with, for example, complex light models.

There has been a recent influx of books in data visualization, which provide material for teaching these topics, for example, those by Cook and Swayne (2007), Murrell (2006 2009), Sarkar (2008), Theus and Urbanek (2009), Wickham (2009a). We also refer the reader to the Wiki (Nolan, Temple Lang, and Hansen 2007b) for links to a variety of new courses in visualization.

In addition, the display of complex or large data, such as network topologies, DNA sequences, and geographic maps, requires new visualization techniques. Furthermore, the way we view graphical representations of data has significantly changed in recent years. Creative and imaginative ways to display complex data using tools such as Google Earth and interactive Web-based technologies are becoming more prevalent. Viewers expect to interact with a graphic on the Web by clicking on it to get more information, to produce a different view, or to control an animation.

5.6 Integrated Development Environments

Our students may be digital natives, but the majority of them are not familiar with basic practices for organizing and carrying out a project involving computing on data. The use of appropriate technologies and clear workflow is important to model and reinforce throughout our curriculum to ensure that these skills are on a firm footing. These topics include: familiarity with file formats, for example, understanding the difference between an HTML file and a Word document; shell tools for managing files and task flow; version control to coordinate work across a team; software distribution, for example, building an R package; and recent developments in reproducible analysis tools, for example, the works by Lenth (2009), Leisch (2002), Gentleman and Temple Lang (2007), Nolan and Temple Lang (2007), Long (2009).

For example, it is increasingly important to understand the shell/command line. It offers an entry point to the command line interface to the operating system, programmatic handling of files, running programs in batch mode, remote login to other systems, and general management of a computational process.

In one approach we use to teach this material, we take a project that students have just completed and step through it showing how we would have carried out the project. We demonstrate how we use the computing environment to do our work, introducing along the way text editors, shell tools, file system concepts, etc. This material is very new to most students but an important component of modern scientific computing for many disciplines.

6. SUMMARY

We believe strongly that the field of statistics is at a crucial tipping point and bold measures of reform are called for in revising the curricula. Modernizing the statistics curricula to include computing in the way proposed here is an issue that needs widespread attention and action. We believe faculty need to indicate to students that computing is an important element of their statistics education, and it must be taught with an intellectual foundation that provides students with skills to reason about important computational tasks and continue to learn about new computational topics. To a large extent, this means learning from the past and challenging the status quo. Instead of teaching similar concepts with varying degrees of mathematical rigor, statisticians need to address what is missing from the curricula and take the lead in improving the level of students' data competence. It is our responsibility, as statistics educators, to ensure our students have the computational understanding, skills, and confidence needed to actively and wholeheartedly participate in the computational arena.

These changes are necessary in order to attract and prepare future statisticians, and to keep pace with the rapidly changing "big science" fields. As the practice of science and statistics research continues to change, its perspective and attitudes must also change so as to realize the field's potential and maximize the important influence that statistical thinking has on scientific endeavors. We agree with SIAM's perspective (SIAM Working Croup on CSE Education 2001) that computational science (which includes statistical computing) needs to be considered "an equal and indispensable partner, along with theory and experiment, in the advance of scientific knowledge and engineering practice."

One challenge to the success of this sea change is that statisticians often have not been taught computing formally, they have not had the opportunity to learn it well, and feel they cannot teach it effectively and so the cycle persists. This is very unfortunate as it means that new students do not have the opportunity to learn it well either. Furthermore, computing has become so much more important in the statistics field than even five years ago that a "just enough" level of understanding of computing is not adequate. We hope to encourage statisticians to break from tradition and take on the challenge.

Now is the time to dramatically rethink our curricula. In the past, some have argued that there is no room for computing in our already packed curricula. While this may have once been true, the growth in data analysis and availability in all sciences and the relative intractability of complex models and methods make computational skills of immense importance in modern statistics. Instead of trying to fit bits and pieces into a crowded curriculum, statisticians must take the opportunity to be bold and design curricula from scratch that embrace new and innovative topics and paradigms for teaching.

The call for change we have made here is not only a call to increase the quantity of computing topics in the curriculum, but includes a change in the topics taught, how they are taught, and how they are integrated with other topics. The ideas presented here are a reflection of our work in designing statistics courses and programs where statistics students attain their potential. Although we have expressed an argument that is quite utilitarian, at the same time it is less end-oriented, instead focusing on computational reasoning. With this approach, we believe that students will be able to embark on new projects and ultimately improve statistics and sciences for the future. We hope these ideas and this discussion will seed the statistics community to make significant changes to how and what we teach statistics students.

SUPPLEMENTAL MATERIAL

Example projects: This document provides brief descriptions of several datasets and simulation studies the authors have used in projects to teach various statistical computing topics. (SupplementaryMaterial.pdf)

[Received July 2009. Revised March 2010.]

REFERENCES

Angel, O., Holroyd, A. E., and Martin, J. B. (2005), "The Jammed Phase of the Biham–Middleton–Levine Traffic Model," *Electronic Communications in Probability*, 10, 167–178. [103]

Barry, D. (2005), "A Conversation With John Hartigan," *Statistical Science*, 20, 418–430. [99]

Brillinger, D., and Stewart, B. (1998), "Elephant Seal Movements: Modelling Migration," *Canadian Journal of Statistics*, 26, 431–443. [104]

Brown, E., and Kass, R. (2009), "What Is Statistics?" *The American Statistician*, 63, 105–110. [97,98]

CAUSEweb (2009), Consortium for the Advancement of Undergraduate Statistics Education, available at *http://www.causeweb.org/*. [101]

Chambers, R. B. J., and Wilks, A. (1988), *The New S Language: A Programming Environment for Data Analysis and Graphics*, Pacific Grove: Wadsworth & Brooks/Cole. [99]

Cook, D., and Swayne, D. (2007), *Interactive and Dynamic Graphics for Data Analysis: With R and GGobi. Use R*, New York: Springer. [105]

DASL Project (2009), The Data and Story Library, available at *http://lib.stat.cmu.edu/DASL/*. [101]

Dimov, D., and Mulloy, B. (2009), Swivel, available at *http://www.swivel.com*. [105]

Friedman, J. (2001), "The Role of Statistics in the Data Revolution," *International Statistics Review*, 69, 5–10. [100]

GapMinder Foundation (2008), GapMinder World, available at *http://www.gapminder.org/world*. [98]

Gentle, J. (2004), "Courses in Statistical Computing and Computational Statistics," *The American Statistician*, 58, 2–5. [104]

Gentleman, R., and Temple Lang, D. (2007), "Statistical Analyses and Reproducible Research," *Journal of Computational and Graphical Statistics*, 16 (1), 1–23. [106]

IBM (2009), Many Eyes for Shared Visualization and Discovery, available at *http://manyeyes.alphaworks.ibm.com/manyeyes/*. [105]

James, D. A. (2009), "RSQLite: SQLite Interface for R," available at *http://cran.r-project.org/web/packages/RSQLite*. [103]

Kaplan, D. (2004), *Introduction to Scientific Computation and Programming*, Pacific Grove: Thompson Brooks/Cole. [98,99]

King, T., Kopf, S., Haenselmann, T., Lubberger, C., and Effelsberg, W. (2008), "CRAWDAD Data Set Mannheim/Compass (v. 2008-04-11)," available at *http://crawdad.cs.dartmouth.edu/mannheim/compass*. [104]

Lahman, S. (2008), "Sean Lahman's Baseball Archive," available at *http://baseball1.com*. [103]

Lange, K. (2004), "Computational Statistics and Optimization Theory at UCLA," *The American Statistician*, 58, 9–11. [104]

Leisch, F. (2002), "Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis," in *Compstat 2002—Proceedings in Computational Statistics*, Heidelberg, Germany: Physika Verlag. Available at *http://www.ci.tuwien.ac.at/~leisch/Sweave*. [106]

Lenth, R. (2009). Statweave, available at *http://www.cs.uiowa.edu/~rlenth/StatWeave/*. [106]

Long, S. (2009), *The Workflow of Data Analysis Using Stata*, Boca Raton: CRC Press. [106]

Mathworks (2009), Matlab, available at *http://www.mathworks.com/*. [100]

Monahan, J. (2004), "Teaching Statistical Computing at North Carolina State University," *The American Statistician*, 58, 6–8. [104]

Murrell, P. (2006), *R Graphics. Computer Science and Data Analysis*, Boca Raton: Chapman & Hall/CRC. [105]

——— (2009), *Introduction to Data Technologies. Computer Science and Data Analysis*, Boca Raton: Chapman & Hall/CRC. [105]

Nolan, D., and Speed, T. (2000), *Stat Labs: Mathematical Statistics Through Applications. Springer Texts in Statistics*, New York: Springer-Verlag. [101]

Nolan, D., and Temple Lang, D. (2007), "Dynamic, Interactive Documents for Teaching Statistical Practice," *International Statistical Review*, 75 (3), 295–321. [106]

——— (2011), *Scientific Computing With Data*, New York: Springer. [104]

Nolan, D., Temple Lang, D., and Hansen, M. (2007a), "Computing in Statistics: Model Courses and Curricula Website," available at *http://www.stat.berkeley.edu/users/statcur*. [99,101,103]

——— (2007b), "Wiki: Computing in Statistics: Model Courses and Curricula," available at *http://www.stat.berkeley.edu/twiki/Workshop/CompCurric*. [103,105]

Peck, R., and Chance, B. (2007), "Assessment at the Program Level: Using Assessment to Improve Undergraduate Statistics Programs," Office of the Dean (COSAM). Available at *http://works.bepress.com/rpeck/1*. [97]

R Development Core Team (2006), *R: A Language and Environment for Statistical Computing*, Vienna, Austria: R Foundation for Statistical Computing. Available at *http://www.R-project.org*. [100]

Ramsey, F., and Schafer, D. (2002), *The Statistical Sleuth* (2nd ed.), Pacific Grove: Duxbury Press. [101]

Sarkar, D. (2008), *Lattice: Multivariate Data Visualization With R. Use R*, New York: Springer. [105]

SIAM Working Group on CSE Education (2001), "Graduate Education in Computational Science and Engineering," *SIAM Review*, 43, 163–177. [106]

SpamAssassin (2007), SpamAssassin Challenge, available at *http://wiki.apache.org/spamassassin/SpamAssassinChallenge*. [104]

Theus, M., and Urbanek, S. (2009), *Interactive Graphics for Data Analysis: Principles and Examples. Computer Science and Data Analysis*, Boca Raton: Chapman & Hall/CRC. [105]

Utts, J., and Heckard, R. (2003), *Mind on Statistics* (2nd ed.), Pacific Grove: Duxbury Press. [101]

Weiman, C. (2008), "Optimizing the University—Why We Need a New Educational Model for a New Century," available at *http://www.scientificblogging.com/carl_wieman/*. [101]

Wickham, H. (2009a), *ggplot2: Elegant Graphics for Data Anaylsis. Use R*, New York: Springer. [105]

Wickham, H. (2009b), "Sections on Statistical Computing and Graphics Data Expo," available at *http://stat-computing.org/dataexpo/2009/*. [101]