Class Proposals for Computing for Statistics

Duncan Temple Lang

January 13, 2006

There are a variety of different approaches and many things that we want to teach the students and have them know when the graduate. Of course, we want them to understand statistical principles. This involves the foundation of classical statistics. We also want them to be familiar with and have, at least, an understanding of modern methodology. We do end up teaching statistical and probabilistic concepts at successive levels of detail/rigor which is a luxury we may not be able to afford. As we refine material via upper division classes, we sacrifice opportunities to teach other material.

Computing – and its many different aspects – is a fundamental aspect of statistics, ranging from simulation, to data collection and cleaning, to exploratory data analysis, to algorithms for model fitting, and reporting results. If mathematics is the language of probability and statistics, computing is the medium. It has historically been taught only tangentially, as supporting material for other classes. And, of course, technology and computing are becoming increasingly important, both as a means for performing data analysis and as a source of interesting data problems. The challenge of large and complex data problems means that students need to be more facile with computation and be able to solve computational problems in order to advance with research. Understanding information technology, both to access data and report results, is an important aspect of the statisticians job. It is something with which our undergraduate and Masters students should be familiar. And lastly, as different aspects of computing evolve, it is important that we, as a community, foster research in the area so as to ensure that the developments in statistical computing over the past 20 years continue in the future.

We must decide where we want the students to know and also what is practical. We want them to be mathematically proficient, computationally capable, to have a strong sense of data analysis and a general understanding of science. Mathematical and computational expertise are a must for every student. We may need to review our admissions criterion to increase the importance of a computing background, perhaps at some price to the mathematical skills. Every student will need to work with a computer, be it simply organizing searches, writing documents, running simulations, performing data analysis and consulting, or developing software for the next generation.

We have to decide how much specialized material we *need* our students to know. For mathematical statistics, there are certain results and techniques that they need to know and some that are specialized and of primary use for those working in mathematical statistical research. Similarly, computing can be taught at several different levels of specialization. Since most computations are and should be done in high-level languages such as Matlab, S and SAS, detailed understanding of low-level random number generation techniques, numerically stable linear algebra algorithms, etc. are not obviously high priority topics when they eliminate teaching other topics. Such understanding is important for those developing new techniques, while more applied understanding will suffice for most students.

Along with good programming and computational problem solving skills, I believe that we need to aim to teach our students how to create software, not simply scripts that they, and only they, can run. Software is structured code that can be used by others. This is is an important and rapidly emerging mechanism for distributing methodological research results to other statisticians and, importantly, to scientists in other disciplines. The results of methodological theses should include software that others can deploy.

An appreciation for the importance of algorithmic design and sensible data structures is important. Students should be able to perform an approximate analysis of an algorithm. They should be familiar with different fundamental algorithms such as dynamic programming, genetic algorithms, acceptance/rejection sampling. They should be able to work with software such as C/Fortran libraries, S and Matlab packages, etc. and utilize these easily. As much as literature review is a necessary skill in traditional research, so is being familiar with available software and algorithms in order to avoid reinventing wheels (with errors).

Much of statistics involves accessing data from different sources, using a variety of different computational techniques and tools to prepare and analyze the data and present results. An understanding of major developments in information technology is important for a statistician for several reasons. It allows them to collaborate with different disciplines using common technology. It enables us to direct and manage non-statisticians in projects that involve diverse data sources and computing needs. And it encourages us to make use of the best available tools for our own computing tasks.

I have included the tentative outline for 5 classes that relate to computing and visualization for statistics. These are:

1. Fundamentals in Statistical Computing.
2. Information Technology,
3. Computational Statistics,
4. Advanced Statistical Computing
5. Data/Information Visualization.

And, of course, there are many other topics that will complement these, such as the courses Nello has been planning and topics in modern computing and algorithmics. Additionally, the information visualization course described below is not inherently a computing class. And, finally, many of these class can and should be cross-listed with other departments such as CS, CSE, (Applied) Math, Geography, Epidemiology, Computational Biology, etc.

These class cover what I consider fundamental aspects of scientific and statistical computing. The material attempts to guide the students through from basic computing to mode advanced usage and theory. The descriptions of the courses of each enumerates more tangible practical aspects of computing. It is important however that the underlying concepts, ideology and thought process be emphasized in the classes and they do not amount to a "how to" for current software which will inevitably become outdated or obsolete. At all times, the focus is on learning how to think about, describe and approach practical aspects of data manipulation and computing for statistical problems. We do not want to teach simply procedure (e.g. point-and-click analyses or how to use particular software), but rather how to think about and understand the technology, and be able to understand issues in computing. At the minimum, we want students to be able to be able to start on problems, and get past simple show-stoppers and be able to recognize the different between a trivial and a serious issue. We want them to graduate to being able to understand how to be able to take a problem from beginning to end. And we want them to be able to understand, compare and critique approaches of others.

The courses span introductory material to advanced and specialized computing. Because most of our students have a limited background in computing, it is possible to combine classes for both graduate and under-graduate students. This is the case, in particular, for the Fundamentals in Statistical Computing, Information Technology for Statistics and Data Visualization classes.

The five courses, when combined with the existing and new classes in mathematical statistics, data analysis and data mining, should provide the basic structure for an emphasis in Statistical Computing for the undergraduate major and also for a specialization in the Ph.D. or Master's programs.

Fundamentals in Statistical Computing

This is a class intended to teach people the fundamental elements of computing for both statistical and general scientific use. It aims to familiarize students with a basic knowledge of programming techniques and practical experience in several programming languages used in statistics and data analysis research and practice. Programming covers basic syntactic aspects of languages, programming style, problem recognition and approaches and software engineering, and debugging principles and approaches. In the course, we will also familiarize the students with different software (e.g. R, SAS, Matlab) so that they should be able to use any of these in other classes, research and consulting.

Students often get distracted with computational aspects of statistical methodology and theory. This course aims to avoid the situation where the students are not sufficiently familiar with software and programming to work and focus adequately in other classes on that class' material.

In some respects, this formalizes and extends material that has been taught briefly in our 390 course. Having this as a formal, required class will help to emphasize its importance, comparable with other more traditional courses. It also will help us avoid students learning this material on their own and frequently incorrectly and insufficiently.

This is a practical course in many respects, but it will discuss conceptual aspects of all the topics and the thought process underlying software and programming rather than simply providing a "how to" or cookbook approach to computing. The idea is to provide the students with a basis with which they can discover additional information and gradually become autonomous. In other words, we are teaching them to learn about statistical software rather than simply teaching the nuts and bolts of current statistical software.

Topics

1. Taxonomy of Statistical Software and programming languages. What are the common tools and what is each one good for.
2. Introduction to the R environment.
3. Basic programming and the R programming language.
4. Developing functions, debugging, packages/libraries and software design.
5. Introduction to Matlab.
6. Introduction to SAS.
7. Basic Graphics.
8. Input and Output for accessing basic data.
9. Unix shells and tools.
10. Batch programming & background tasks.
11. Perl.
12. Basic inter-system interfaces (shell, C, Fortran, Java, Perl).
13. LaTeX, Word processing and reproducible results.
    Sweave, XML.

In order to make the concepts taught in this class concrete, we will want to have practical examples and projects on which the students can work. Given that most of them will be at an early stage in their education, they will have had little exposure to statistical methodology. Simulations and intuitive methodology (e.g. CART) can be used. We will have the students work on projects using the different languages so that they appreciate the relative merits of the different software available to them.

Information Technology for Statistics

Computing technology is changing rapidly and both software and data are available to us in many different forms. We need to be able to access complex data that comes to us not in rectangular tables, but in "raw" forms such as documents and citations, log files, DNA sequences, gene annotations. At the same time, the data is increasingly part of a dynamic process being updated frequently, and available to us from diverse engines such as relational databases, Web and mail servers, monitoring devices, and software itself. Along with data, software is increasingly complex and the number of ways in which it is accessible is increasing: C libraries, Java packages, Perl modules, HTML forms, Web services (e.g. SOAP). The potential for interesting and rich interactions for a statistician is greatly increased if she has a good knowledge of information technology and can readily understand the components within an Information Technology (IT) department, and can also access data and functionality from the different elements of the information system. This is increasingly becoming an important, if not essential, part of the practicing statistician's skills.

This course aims to introduce many of the important and emerging tools in the information technology domain in the context of their use in statistical practice and research. It builds on a knowledge of a high-level computing language such as R or Matlab and discusses software development for the data analysis process (collecting, preparing and analyzing data, and reporting results).

We cover text processing, accessing and creating relational databases,

1 Topics

- How computers work.
- Taxonomy of Scientific Computing Languages. Operating systems, interpreted and compiled languages, networks and communication.
- Review of R
- Software Development
  Writing functions, debugging
  Software reuse
  Packages and Modules
  Object Oriented Programming
- Text Processing
  Basic stream (Unix) tools
  Pattern matching and Regular Expressions
  The role of Perl in text manipulation.
- Relational Databases Management Systems
  The relational model
  Database Query Language - SQL
  Database Schema design.
- Structured, Self-describing data
  XML and parsing
- Integrating Software Components
  Remote Procedure Calls using SOAP and DCOM.
  Calling Perl, Matlab, etc. from R.
- Statistical Reporting
  Generating on-line statistical content.
  HTML, graphics

Computational Statistics

This is a class that teaches what is traditionally called Statistical Computing. It is, however, more accurately described as computational statistics and relates to algorithms and numerical computing. The goal is to give students a foundation in common computational techniques and algorithms that are used in the implementation of statistical methodology. This is an important part of statistics in todays world as the implementation of an approach is important in its efficacy. Many new methods involve combining different algorithms, and being familiar with the different approaches and when they work is important. When is one approach likely to work better than another? What are the differences? What software is available in different packages?

The primary purpose of the course, as Ken Lange puts it, is to allow students to create their own software, and also to understand the components and underpinnings of existing software. We hope that students will be able to contribute robust and accurate software to support their research and also to be able to critically understand the techniques used in existing software. This course aims to provide the students with a suitable introduction to this topic.

This is basically a graduate course. It is an important class for Ph.D. students, possibly taken in their second year, but not necessarily essential for all students. A scaled-down version of this class could be taught at the upper-division, undergraduate level. This would be more to explain how things work and under what circumstances they work well or not.

In ways, this course overlaps with elements of Nello's 'Theory of Computation' class. Elements of this course are also taught in other departments. For example, Math 258 is Numerical Optimization. (e.g.

2 Topics

- Representation of Numbers and computation errors, accuracy.
  Limitations of computers.
  Errors when computing sums, means, variances.
  Reorganizing computations: e.g. factorials versus gamma function.
  Errors in software (e.g. Excel, Peter McCullough's article in the American Statistician).

- Elementary Numerical Analysis.
  Matrix algebra. Singularities. Ill-conditioned matrices.

- Algorithmic complexity.
  Big O notation, calculations, etc.

- Decompositions.
  QR, SVD related to statistics.
  Eigenvalues.

- Random Number Generation (RNG).

- Simulations and Computer Experiments.

- Resampling
  Cross-Validation, Bootstrap, jackknife.

- Fast Fourier Transform (FFT).
  Convolutions. Time-series.

- Optimization
  Traditional algorithms: Newton-Raphson, Nelder-Mead, Gradient
  Descent, etc.
  Simulated Annealing.
  Genetic Algorithms.

- Markov Chain Monte Carlo.
- Computational Aspects of Expectation-Maximization (EM) algorithm.
- Additional Topics:
  - Dynamic Programming. (Could be moved to basic programming intro class.)
  - Sampling schemes (e.g. importance sampling).

3 Pre-requisites

Statistical Computing & Software course or experience in Matlab or S.

4 Possible Textbooks

- Numerical Analysis for Statisticians, Kenneth Lange.
- Computational Statistics, Geof Givens and Jennifer A Hoeting (To appear in 2005).

The following focus more on statistical methodology (e.g. clustering) rather than general elements that are used in many different methods, e.g. optimization. These topics are probably better left to

- Elements of Computational Statistics, James Gentle.
- Computational Statistics Handbook with Matlab, Wendy & Angel Martinez.

Older books are

- Statistical Computing, Gentle & Kennedy, 1980.
- Statistical Computation, Maindonald, 1984.

Advanced Statistical Computing

This class is intended for covering intermediate and advanced topics in statistical computing and information technology. Since computing is a very dynamic, active and fast-changing area, the course will also change. Additionally, as our students become more advanced computationally, we can focus on more interesting topics rather than using this class to merely raise the level of basic computational literacy.

- Software design and architecture for statistical computing problems.
  How to write functions, classes, etc. for flexibility, efficiency, reuse.
- Debugging.
- Efficiency.
  Basic concepts in efficiency for large datasets and computationally intensive tasks.

One version of the class focuses on High Performance Computing, i.e. dealing with large datasets and computationally intensive methods. In the course we hope to provide a mix of formalism, infrastructure, tools and applications dealing with efficient, high performance computing related to data analysis, simulation, scientific computing and visualization. There will be a mix of surveying the topic and providing practical approaches to problems. The aim is to provide people with an understanding of the issues in the area of resource-constrained computing problems and to address practical solutions and understand their characteristics. The course focuses on using high-level languages and the associated tools while mixing these with other facilities and languages to perform computations.

The course starts with some examples of computations that consume resources and are infeasible. We will then explore ways to measure and understand the use of resources that make these examples infeasible and explore different approaches to reducing the consumption of these resources. This will be done in S and Matlab and compare the two for flexibility and efficiency. The techniques we will explore will include general and simple approaches such as constant-folding/invariant extraction from loops, pre-allocation of space, and elementary rewriting of code. We will also discuss algorithms and reorganizing computations to speed them up or simply make them feasible. These include taking advantage of the particular problem to rewrite the mathematical computations in an equivalent way, and also to rewrite the basic computations to do them in different order and cache results to save computations. We will also discuss how to estimate/approximate the resources that are needed for a particular task (i.e. memory and time/cycles) using both theoretical computations, and experimental approaches including simulations and profiling. This requires a basic understanding of algorithmic complexity.

We will discuss examples of algorithm-specific approaches to improve efficiency which illustrate the general technique of reorganizing the computations. We will also discuss data reduction techniques such as sampling and how they can be used to get approximate answers which can be used as-is or as starting points for iterative algorithms on the entire data.

Having discussed approaches for efficient computation in high-level languages, we will discuss the integration of lower-level, high-performance languages. C/C++, Fortran and Java are commonly used languages in scientific computing that provide high performance because they are strongly-typed, compiled languages. We will discuss how to use software developed in these languages from within S and Matlab. This covers aspects such as writing interface code, compiling, linking and loading the native code, and debugging when things go wrong. This will be taught at two levels: one for this with little knowledge of the native language (e.g. C), and another that exploits knowledge of these languages.

We next talk about distributed and parallel computing. We present the essential framework and concepts. We describe the problem of synchronization and the approaches and tools used to resolve this. Next, we move to using distributed computing in R. This involves the SNOW and either of the Rpvm or Rmpi packages, and even higher-level approaches using Common Object Request Broker Architecture (CORBA) and Distributed Common Object Model (DCOM).

Distributed computing has further complexities in addition to the synchronization issue. Random number generation across numerous interacting agents requires an understanding of the different possible models and semantics and reproducibility.

We will look at using databases to manage data and perform some computations outside of R and Matlab and how this can affect efficiency. We will also look at block algorithms which avoid the need to hold all the data in memory at any one time. These lend themselves naturally to use in the important area of streaming data.

1. Basic efficiency approaches in S and Matlab.
   Vectorization, looping, pre-allocation of memory, apply, outer.

2. Identifying sources of potential inefficiency/efficiency
   Understanding computers resources and how they are managed
   Measuring resource usage
   Understanding when and where to optimize algorithmic complexity, profiling, memory management.

3. Examples of rewriting computations more efficiently.
   i.e. reorganizing the mathematics or basic procedural steps to take advantage of application-specific efficiencies.

4. Sampling approaches for data reduction.
   Approximate answers to questions with inherent errors.
   Good starting values.
   Limitations - erroneous sampling strategies.

5. Integrating C/C++/Java/Fortran code into S or Matlab.
   Understanding when to use these languages
   Dynamic Loading.
   The .C/.Call/.Fortran/.Java interfaces.
   Compilation, linking, make utility and configuration for portability.
   Support from R and Matlab compilation mechanisms.

6. Creating statistical software.
   The R package mechanism.
   Approaches for creating reusable libraries.

7. Integrating other languages.
   Perl, Python, etc.

8. Using client-server model for computation.
   Managing data and some computations in relational databases.

9. Paradigms for parallel and general distributed computing
   processes/threads, synchronization, atomicity, deadlock, semaphores/mutexes, events and asynchronous computing.
   Fault tolerance.

10. Distributed computing in R
    SNOW, Rpvm and Rmpi, Condor.
    CORBA and DCOM.

11. Simulation and random number generation (RNG) in distributed computing

12. Block and record-at-a-time algorithms.

13. Data streams.

Additional Specialized Topics

- Parallel programming languages
  e.g. Fortran 90

- Visualization for large datasets.

Pre-requisite: familiarity with a programming language, either a compiled (C/C++, Java, C#) or a higher-level interpreted language (S/R/S-Plus, Matlab, Perl or Python).

Students will be expected to propose and implement a project involving aspects of efficient computing. This can be part of their research. Some potential topics include

- XML streams
- Web logs
- network data (SNMP, routing, packet sniffing, etc.)
- simulation and SPRNG
- Algorithms: MCMC, CART, optimization, image processing
- interfacing to MPI/PVM/CORBA/DCOM
- experimental design for operating characteristics of a piece of software
- algorithmic analysis
- approximate queries and estimation algorithms
- optimization approaches for a code-base of interest
- evaluation of the efficiencies and inefficiencies of good software practice.

Data Visualization

Graphics and visualization plays a vital role in statistics. It is used when collecting and cleaning data to identify anomalies and outliers. It is used in the early stages of modeling as exploratory data analysis (EDA). It is used in model diagnostics and helps to motivate refinements of the EDA-modeling process. And lastly it is used in presenting the results of the analysis, either statically or dynamically as part of the overall decision making process. In spite of its importance in all aspects of the statistical process, visualization is not explicitly taught as part of the statistics curriculum, and students are expected to "pick it up" as required.

This class aims to provide a framework with which students can critique, compose and create graphics that usefully display information from data. We will cover not only the mechanics of creating graphics, but what makes a good display and why. They will learn about the fundamental elements of visualization, primarily cognitive perception and different aspects such as shape, size, texture, color models. We will discuss the standard types of displays used in statistics and when they are useful and appropriate. We will also look at creative and imaginative ways to display more complex data such as network topologies and throughput, DNA sequences, computer code, and geographic maps, using both static and dynamic approaches.

This is both an upper-division undergraduate class and graduate class. This class can be taught to both graduates and undergraduates at the same time. The students are expected to read about the theory of perception, research examples of different aspects of visualization, and become familiar with software to be able to create different displays and discuss their merits. Students are encouraged to experiment and explore different displays by creating them. Each student will be expected to work on a significant, innovative project that either explores approaches for visualizing a particular dataset, (re)constructing or improving complex displays, or providing a general framework (with software) for visualizing a particular type of data. If the class is taught at the graduate level, we can assign more reading of research papers on topics such as experiments on perception, information design, etc.

This course should appeal to many students in different disciplines, specifically any field that involves the presentation of data.

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.

5 Content

- The role of graphics
  comparison with, e.g. tables with respect to lookup, comparison, etc.
  Purpose of graphics in a paper.
- Anatomy and language for graphics.
  Wilkinson's grammar of graphics.
- Cognitive Perception.
  Retinal variables, optical illusions.
- The collection of common display types in statistics.
  bar charts, scatterplots, time series, parallel coordinates, mosaic plots, coplots, trellis/lattice plots, ...
- Annotations
  legends, etc.
- Critique of common displays.
  Examples of good and bad graphics in public forums.
  Examples of software creating low quality displays.
- Rules for good composition.
- Software for static graphics
  R and Matlab's graphics models and facilities.

- Graphics for Exploratory Data Analysis and presentation.
- Interactivity and Direct manipulation graphics
  Models
  Software: GGobi, Orca, iPlots, OpenGL & rgl, Matlab, Java applications.
- Maps.
- Approaches to visualizing “large data”.
  hexagonal binning, ...
- Aspects of color and texture.
- Visualizing non-traditional data
  e.g. text, software, etc.

5.1 Special Topics

- Graph and Tree layout (i.e. for nodes and edges)
  algorithms and examples of their use.
- Geographic Information Systems (GIS).
- Grand Tour and data-driven animation.
- Graphical User Interface (GUIs). Design and implementation.
- Streaming data and its challenges.
  what are we trying to visualize
  issues of changing scales and resolution
  connection to adaptive models.
- Virtual Reality.

6 Textbooks

- Visualizing Data, William S. Cleveland,
- Elements of Graphing Data, William S. Cleveland.
- The Grammar of Graphics, Leland Wilkinson.
- The Visual Display of Quantitative Information, Edward R Tufte.

6.1 Additional Material

Information Visualization, Perception for Design

7 Examples

Basic datasets, longitudinal data, DNA Sequences, Computer Networks, Software visualization, Visualizing Parallel computing, Census data, Spatial Environmental data.