A Programming Course in Bioinformatics for Computer and Information Science Students

Russ B. Altman
Section on Medical Informatics
SUMC, MSOB X-215
Stanford, CA 94305-5479
altman@camis.stanford.edu

John Koza
Department of Computer Science
Margaret Jacks Hall
Stanford, CA 94305-2140
koza@cs.stanford.edu

Abstract

We have created a course entitled "Representations and Algorithms for Computational Molecular Biology" with three specific goals in mind. First, we want to provide a technical introduction for computer science and medical information science students to the challenges of computing with molecular biology data, particularly the advantages of having easy access to real-world data sets. Second, we want to equip the students with the skills required of productive research assistants in molecular biology computing research projects. Finally, we want to provide a showcase for local investigators to describe their work in the context of a course that provide adequate background information. In order to achieve these goals, we have created a programming course, in which three major projects and six smaller assignments are assigned during the quarter. We stress fundamental representations and algorithms during the first part of the course in lectures given by the core faculty, and then have more focused lectures in which faculty research interests are highlighted. The course stressed issues of structural molecular biology, in order to better motivate the critical issues in sequence analysis. The culmination of the course was a challenge to the students to use a version of protein threading to predict which members of a set of unknown sequences were globins. The course was well received, and has been made a core requirement in the Medical Information Sciences program.

1. Introduction

At Stanford University, students can become active in molecular biology computing through numerous routes. The departments of computer science, medical information science, electrical engineering, structural biology, biochemistry, genetics, and mathematics all have students who are currently engaging in research that can be considered within the scope of computational molecular biology. The ways in which these students choose their research projects are, of course, quite different. The students in the biological sciences are usually part of a group that is motivated to solve a particular problem, and computer technology has been identified as a key component of the solution. The students in the technical fields have a much more diverse set of paths. Sometimes, they have had an interest in biological problems, and have sought advisors with computational efforts. Other times, the algorithms being used by a group are of particular interest to the students. Some students are hired simply for their programming skills.

We have noted that the more serendipitous pathways to computational biology taken by engineering and computer science students are not without risks: (1) The "big picture" of the research may be missed by these students, and may lead to substantial communication problems in the course of the work. This often involves a basic understanding of the biological vocabulary, but can also involve understanding the performance goals of a program. (2) There is wide variation in the degree of preparation these students have for working with biological data. The inherent noise level of biological data is often unexpected for students familiar with more restricted, formalized engineering domains. The lack of previous exposure can also lead to attempts to reinvent the wheel. (3) Many students with potential interest in biological computing as an application area for new algorithmic methods have no good way to be introduced to the computational biology literature and previous results. They often select application domains that are more familiar, and apply new algorithms to familiar computer science topics such as string matching, code parsing, virtual reality. Given an appropriate introduction, they might instead choose areas such as sequence analysis, sequence-structure correlation and three-dimensional structure modeling and analysis.

In order to address these problems, we created a course that was specifically targeted to non biological computer science and information science students. The title of the course was "Representations and Algorithms for Computational Molecular Biology," and was offered in the spring quarter, 1995. The course was widely advertised especially to graduate students and undergraduates in computer science and medical information science. The course required significant computer programming, some use of off-the-shelf molecular biology computational tools, and use of WWW resources. We stressed that the goal of the course was to prepare students for building such tools, not necessarily being "power users" of such tools. We also stressed that extensive knowledge of biology was not a requirement for taking the course, but that an interest in biology was essential. In the remainder of this report, we discuss how we addressed three critical issues: (1) teaching "just enough" biology, (2) selecting programming assignments, and (3) putting together a lecture schedule. Finally, we will summarize our evaluation of the course, as well as the student feedback we received. This course was taught by two faculty members. One of us is in the Section on Medical Informatics (part of the Department of Medicine, described in Appendix III of the supplementary material included with the electronic proceedings) with a courtesy appointment in Computer Science. The other of us is a consulting faculty in both the Department of Computer Science, and the Program in Symbolic Systems. The course had a single teaching assistant, who was a senior graduate student in computer science, focusing on biological applications in his dissertation work.

1.1 Teaching "just enough" biology

We stated from the start that knowledge of biology was not required, but interest in biology was. The first lecture was an introduction to biology and the role of information sciences. Starting from the definition of entropy, we argued that the processes of life are a constant struggle against entropy. Specifically, having established the distinction between inside and outside, an organism must use energy to maintain that distinction, and must develop strategies for gathering, storing, and using the energy for that purpose. We argued that the basic tools available to organisms are the ability to store information in DNA and the ability to effect function with protein (acknowledging the possibility of interesting variations). Using this paradigm, we introduced the alphabets of DNA and protein, and explained the ways in which some DNA sequences specify protein sequences, and others specify "control information" necessary for timing and synchronizing molecular expression and physical associations. We included sections of a standard biochemistry text book in the course reader (see Appendix I in the supplementary material included with the electronic proceedings).

In the introduction, we made clear to the students that they would learn biology in a skewed manner during the course. Specifically, they would learn little about how biological data is collected experimentally. In order to further set expectations, we stressed that they would not learn how to use the available molecular biology software, and would not be working on many state-of-the-art biological problems. Instead, they would be exposed to the basic representations used for DNA (sequence and the issue of pairwise and multiple alignments) and protein (sequence and analysis techniques for three-dimensional structure).

In subsequent lectures, we introduced a small amount of relevant biology before each lecture in order to motivate the techniques being explained. Thus, for example, in one lecture, we explained the functional characteristics of globin molecules and their interaction with heme and oxygen, in order to provide background necessary to explain the importance of conservation of HIS and GLY residues (for multiple alignments of globin sequences), and the locations of the conserved helices (for evaluating Hidden Markov Models (HMMs), and for explaining secondary structure prediction). The result was that our students had a very uneven understanding of biology, with a detailed understanding of the sequences and three-dimensional structures of globins, but having never heard of important metabolic pathways or mechanisms of gene regulation.

We were very sensitive to the problem of using biological jargon during the course. Each instructor made a conscious effort to monitor the other during lectures. Students were encouraged to interrupt and ask for clarification of apparent jargon with a very low threshold. We also offered a "virtual email server" in which students could ask either of the instructors for a quick email definition of a concept at any time. A suggestion was subsequently made that we create an on-line world-wide web glossary. Although we have done this for other courses, we did not create a WWW glossary for this course.

1.2 Assignments and Projects

During the course, we distinguished between assignments, which were weekly and designed to take 1 to 3 hours, and projects, which were more substantial and required up to twenty hours. The assignments were designed to teach the students how to access biological data, how to perform basic manipulations, and to test out ideas introduced in class. The projects were designed to teach the students how fundamental algorithms in computational molecular biology work, how they may fail, and in what ways they are sensitive to errors in the data. Although the initial assignments were oriented towards using off-the-shelf software, the later assignments and projects all required significant programming. The students were allowed to use any programming language of their choice, and we graded the code by general examination of its quality, and by the quality of the results on the test examples we provided. We summarize here the assignments and projects in the order in which they were assigned (these are available at anonymous ftp://camis.stanford.edu/pub/altman/mis214 or by request to authors).

Assignment 1. "Gathering Biological Data from the Web" Students were given a script in which they were asked to navigate on the WWW to the major biological databases and retrieve a DNA sequence, protein sequence and protein structure for the same molecule, sperm whale myoglobin. They were asked to perform a network BLAST search to pull out other myoglobin sequences as well.

Assignment 2. "Investigating 3D structure of Myoglobin" Students were asked to process the Protein Data Bank (PDB) files for sperm whale myoglobin for input into the Kinemage program, which is available for public use (Richardson, D. & J. Richardson, 1992, "The kinemage: A tool for scientific communication." Protein Science **1**: 3-9.). They were asked to make distance measurements between key points, estimate the location of secondary structures by eye, and look at the geometry of the oxygen binding site. They were also asked to highlight hydrophobic and hydrophilic residues separately in order to demonstrate the "hydrophobic amino acids in, hydrophilic amino acids out" principle.

Project 1. "Dynamic Programming for Sequence Alignment" In the first project, students were asked to encode the standard dynamic programming algorithm for aligning sequences (for both local and global alignment). They were given a standard input format, and a set of test sequences to align. They were also asked to pull two random sequences from Genbank and align them to get a feeling for how such alignments appear and score.

Assignment 3. "Investigating the 3D structure of Hemoglobin and Superoxide Dismutase" In order to introduce the ideas of multisubunit proteins, and to illustrate non-alpha helical secondary structure, we chose PDB files 4HHB and 2SOD. In a manner similar to Assignment 2, but with less explicit instructions, we had the students visually analyze these structures, and make measurements of distances, bond angles and dihedral angles. We also asked them questions that allowed them to focus on the important functional residues in both proteins.

Assignment 4. "Computing Dihedral Angles" The students encoded a subroutine for computing dihedral angles, and applied this subroutine to the structures of myoglobin and superoxide dismutase. They then plotted the resulting phi and psi angles to produce Ramachandran-like plots for each protein, and were asked to identify the helical and beta regions of these plots. They were asked to speculate on how structure might be predicted given a training set of sequence-structure pairs using dihedral angles.

Project 2. "RMS comparison of structures" Whereas the first project was oriented toward sequence analysis, the second project was geared towards comparing three-dimensional data using the root mean squared (RMS) distance criteria. Students encoded a program that took two sets of points as input, and produced the rotation and translation necessary to superimpose them optimally. The students were supplied code for the singular value decomposition, and were instructed to work from an original paper describing the method (Arun, K. S., T. S. Huang & S. D. Blostein (1987). "Least-squares fitting of two 3-D point sets." <u>IEEE Transactions on Pattern Analysis and Machine Intelligence</u> **9**(5): 698-700). They computed the RMS distances between a number of globins, based on alignments generated using the code written for project 1, but with a variety of different gap penalty schemes and with different scoring matrices. They were also asked to superimpose some globins using a subset of "core" helices in order to demonstrate the dependence of RMS on the set of atoms used for the superposition. They then produced an optimal structural alignment and compared this with the best sequential alignment.

Assignment 5. "Distance maps and pattern recognition" The students encoded a subroutine for computing distances, and prepared a plot of all alpha carbon distances less than 6 Å for myoglobin and superoxide dismutase. They were asked to highlight patterns in the plot corresponding to alpha-helices and parallel beta sheets. They were asked to describe how antiparallel beta sheets might appear as well.

Assignment 6. "Computing with Hidden Markov Models" The students were given a simple Hidden Markov Model for a toy system, and were asked to compute the probabilities of various sequences given the model. They were also asked to speculate on the effect of various changes in the model on the probabilities.

Project 3. "Threading protein structures and predicting structure" In an attempt to bring together the concepts learned in the first two projects, the students were asked to create a program for determining if a given sequence was a globin or not. They were given the environmental parameters as reported in Bowie et al (Bowie, J. U., R. Luthy & D. Eisenberg, 1991, "A Method to Identify Protein Sequences That

Fold into a Known Three-Dimensional Structure." Science **253**(July 12): 164-170.), and were asked to use the code developed for Project 1 to align the new sequence with the environmental string corresponding to myoglobin. In order to test their code, they were given 4 sets of proteins: myoglobins, non-myoglobin globins, non-globin all alpha helical proteins, and non-alpha helical proteins. They computed the best score by threading and by dynamic programming for each group of proteins to establish the background rates. They were then given 10 unknowns that contained 3 globins, including some distant globin relatives, and were asked to make predictions about globin vs. nonglobin for each sequence, along with a justification.

Our course was taught in one quarter, which lasts about 10 weeks. Because of time constraints, we were unable to have the students do as many assignments and projects as we would have liked. Specific areas that we missed, and may try to address in future courses included secondary structure prediction (computing Chou-Fasman-like parameters), identifying consensus sequences, fragment reassembly, and the actual computation of threading parameters (as opposed to providing them to the students). Each of these topics was covered in the lectures, but was not covered in the assignments.

1.3 Lecture Content

The lecture topics for the course were selected in order to provide an introduction to the major issues, and then to allow particular interest areas of the instructors to be stressed. Thus, there was an emphasis on structure during the course, and certain areas of sequence analysis (especially the more formal algorithmic issues) were not covered. It is important to note that there are two other related courses taught at Stanford: one of them is taught in the department of biochemistry, and introduces students to the concepts of computational molecular biology, but stresses the use of existing tools. The other course is taught in the department of mathematics, and stresses the basic statistical and algorithmic issues involved in sequence analysis, particular dynamic programming and other alignment techniques. For this course, eleven lectures were given by the core faculty (marked with * here) and the rest were given by local guest lecturers. Each lecture was 75 minutes. The topics of each lecture, in order given, are summarized here. The relevant readings are given in Appendix I in the supplementary material included with the electronic proceedings.

*1. *Introduction to representations and algorithms in computational molecular biology*. We introduced the processes of life as a battle against entropy, fought mainly by developing strategies for gathering and using energy from the environment. We introduced DNA sequences as a method for succinctly encoding information both about three-dimensional molecules, and also about the control systems necessary for timing and development. We introduced protein sequences and structures as complex phenomena chosen from a large search space because of their special characteristics.

2. **Sequence alignment: basic dynamic programming*. After demonstrating the need to compare sequences of proteins and nucleic acids, and discussing the concept of edit distance, we explained the basic dynamic programming algorithm for aligning sequences with affine gap penalties. A small example was worked out by hand, and students were warned that they would be writing a general purpose program as a project. The variations of local and global alignment were covered following the logic of Gribskov and Devereux (Gribskov, M. & J. Devereux (1992). Sequence Analysis Primer. <u>UWBC Biotechnical Resource Series</u>. R. R. Burgess. New York, W. H. Freeman & Co.**:** 124-137.).

3. **Protein structure and structure-sequence relationships: myoglobin as case study.* Using Kinemage software, the students were guided through the structure of myoglobin as the first case study of the relationship between sequence and structure. The concept of a protein as a small computational machine was introduced (and elaborated upon in the sixth lecture). The basic physical properties of amino acids, and the nature of the folding problem were discussed.

4. **Basic computations on 3D structure: distances, bond angles, dihedral angles.* In order to provide technical followup to the previous lecture, this lecture focused on the detailed geometry of protein structures, and the constraints placed on this geometry by nature. Specifically, the calculation of distances, bond angles and dihedral angles was discussed, as well as a look at the empirical distributions of these quantities over sets of protein structures. This allowed introduction of alternative representations for structure, other than three-dimensional Euclidean coordinates, including concepts of internal coordinates (backbone phi, psi angles), distance maps, and object-based coordinate systems. The computational advantages of each were discussed.

5. *The combinatorics of the protein folding problem, and simplifying assumptions.* This lecture focused on methods for managing the large computational search space when predicting protein structure. Lattice models for protein backbone structure were discussed, as well as methods for exhaustive enumeration of conformations. The use of constraints to filter unlikely structures was discussed, along the computational complexity of such operations. The goal of this lecture was to give the students a feeling for the difficulty in predicting protein structure *de novo*, and the need for algorithms which give other clues, based on sequence analysis and the analysis of related structures.

6. **Protein structures as mini-computers: hemoglobin.* This lecture elaborated on the idea of a protein complex as a small computer. After discussing the main features of the hemoglobin tetramer, and the mechanisms of cooperativity, the students were introduced to a series of mutations, along with the (predictable) functional results. In this way, the function of proteins was shown to follow from sequence, and to be modifiable by sequential modifications.

*7. *Analysis of sequences: composition, coding regions, intron/exons, reading frames.* Using a demonstration version of the GeneWorks™ suite of sequence analysis programs, the students were introduced to the range of computations that occur on sequences, and the types of questions that biologists typically ask of a sequence. Sequences were analyzed for amino acid composition, the search for open reading frames, and the locations of introns and exons. The algorithms used for these computations were discussed, as were the critical assumptions underlying them.

8. Alternative representations of sequence: profiles, regular expressions, and belief networks. Having previously introduced the basic statistics of sequences, this lecture focused on more subtle representations of sequence including profiles, regular expressions and belief networks. The assumption of independence of columns was discussed, along with methods for measuring the entropy of an alignment, and methods for detecting covariation or correlation using entropy measures and probabilistic measures.

9. Simulating molecular dynamics of proteins. The basic theory behind simulating macromolecular dynamics was presented, along with the fundamental energy equations, discussion of how they are parameterized, the importance of modeling solvent, and the reproducibility/reliability of the results. An animated molecular dynamics movie was shown.

*10. *Processing distances algorithmically and graphically.* Given the abundance of distance information from both experimental and empirical sources, this talk discussed the ways in which distances can be processed using the theory of distance geometry, constraint satisfaction and restrained molecular dynamics. The relative strengths and weaknesses of these approaches were discussed.

*11. *Identifying transmembrane proteins with genetic algorithms and genetic programming.* The principles of genetic algorithms and genetic programming were introduced, with an emphasis on their particular suitability to biological problems. As an example, the evolution of a program to predict transmembrane domains in proteins was reviewed in detail.

12. Hidden Markov models and multiple alignment. After introducing the basic issues and difficulties with multiple alignment, this lecture focused on the use of HMMs to model biological sequences. The basic data structures, and learning algorithms for HMMs were reviewed, along with some successful biological applications to the globins.

*13. *Extracting non-atomic features from protein structure.* Moving away from purely atomic representations of structure, this lecture focused on looking at protein structures as ensembles of biophysical and biochemical features. The algorithms for defining these features, as well as for detecting significantly conserved features in an ensemble of similar structures were discussed.

14. Computing molecular surfaces and molecular volumes. The essential computations involved in computing molecular surfaces using standard methods were reviewed, as were the basic algorithms for computing Voronoi volumes for molecular structures. The application of these algorithms toward understanding structural docking and molecular interactions was discussed.

*15. *Secondary structure prediction algorithms: Chou-Fasman, GOR, and neural networks.* The basic problem of predicting protein structure was revisisted, now with an emphasis on detecting secondary structures. The basic strategy used by the most common algorithms was reviewed. Secondary structure predictions for globin sequences were sent off in real-time during the class to a number of servers, and the results were compared and contrasted.

16. Genomic computing: fragment assembly. The low level issues involved in compiling the human genome were covered, including the problem of base calling from sequencing machines, the general strategy of sequencing by hybridization, and the basic algorithms used to solve these problems. Special stress was placed on the problems that arise with many basic algorithms in the context of real data that is inaccurate and contains lengthy repeats.

17. Minimum description length principle and constructing evolutionary trees. The problem of constructing evolutionary trees was presented as a problem of constructing a tree with the minimum description (or message) length (MDL). The relationship of MDL methods to Bayesian methods was discussed, and quantitative algorithms for assessing the similarity of sequences using the MDL principle were introduced.

18. Representing, storing and computing with metabolic path information. The critical issues that arise when creating biological databases and knowledge bases were introduced in the context of storing information about metabolic pathways. The value of internet-based tools, and the methods used to design and create them was discussed. A real-time demo of the EcoCyc system for browsing knowledge about E. Coli genomic structure and metabolic pathways was included.

*19. *Constraint satisfaction techniques for computing structure from constraints.* The concept of a constraint satisfaction network, low and high order consistency checks, and their applicability to structure modeling was introduced. As an example, the construction of molecular models of large ribonucleoprotein complexes was considered, along with the use of constraint satisfaction operators to create a representative set of all conformations.

The readings associated with these lectures are given in Appendix I. In general, one or two relevant articles were provided for each lecture. These were bundled together in a course reader that was copied (after obtaining all necessary copyright permissions, a service provided by the copy facility) and sold, at production cost (around $50), in a two volume set (a total of 426 pages).

2. Results & Discussion

The class was announced relatively late in the academic year by electronic mail (and did not appear in the course catalog), but generated 50 attendees during the first three lectures. Four senior faculty from Computer Science, Engineering, and Genetics attended these early lectures, and expressed interest in auditing the course. In addition, a number of post-doctoral fellows from the biological sciences attended these and a fair number of additional courses. In the end, there were 24 students who took the course for a grade. Of these, 6 were graduate students in Medical Information Science, 7 were graduate students in computer science, 6 were undergraduates in computer science, and the remainder were graduate and undergraduate students in engineering. The final assignment in the class was an anonymous questionnaire in which we asked students to rank the importance and value of the lecture topics, as well as the homework assignments. Appendix II in the supplementary material included with the electronic proceedings contains the detailed table of student responses.

By far the most positive lesson learned from teaching this course was the great amount of interest in this field among both graduate and undergraduate students. We initially feared that our late announcement (the course did not appear in the regular printed course catalog) would lead to very low enrollment. Our initial target enrollment was 12-15 students, and so we exceeded this target by almost a factor of two. There were a large number of undergraduates who were not able to fit the course into their schedule because of the late announcement, who showed interest in taking the course at its next offering. Many students see biocomputing as a growing application area, where they expect jobs to be available. As parts of computer science shift focus from fundamental research and defense applications, the opportunities within medicine and biology are receiving increased attention. The attraction of biology is that there are mappings from all the traditional applied computers science subfields (for example, graphics, machine learning, user interfaces, data structures or graph-based algorithms) that allow interests in any of these subfields to be pursued.

In order to limit the amount of biology that we had to teach, we concentrated on a few molecular systems. In particular, we studied the globins, particularly myoglobin, as the primary example for all our sequence and structure analysis assignments, as well as the focus of many of our lectures. This strategy was successful overall, although it did lead to some comments by students that they were getting tired of myoglobin by the end of the class. However, the clear benefit of this approach was that the students had seen the sequences and structures of the globins using in multiple ways during the course, and never had to learn additional biology in order to understand the constraints on algorithms that came from the biology.

The students definitely enjoyed the “hands on” programming flavor of the course, although not without some qualifications. The programming assignments required (for some students) a fairly large amount of “busy work” parsing through data bank files and cutting and pasting data into standard formats. We allowed students to program using any language of their choice. It was our clear impression that students working in higher level languages such as Lisp, Prolog, MatLab, and Perl had many less problems with writing support code than students working in C or C++. This was highlighted in one case where a student using MatLab was able to encode a root mean square fitting routine in less than 30 lines of code! These observations underscored our belief that computer science departments need to ensure that students learn to be productive in at least one programming environment that is geared towards prototyping, and which supports high level operators. In order to assist students in the future, we may provide support code for reading data files into C, or else create data files that are more easily parsed.

We were disappointed, but not surprised, that many students did not complete the assigned readings. We gathered them together in a single course reader in an attempt to minimize the barriers toward reading the primary literature. However, the relatively large programming load distracted most students from the readings, especially toward the end of the course, as shown in Appendix I. In general, the students read those sections of the reader that were required for successfully completing the programming projects. Our difficulty in creating this reader, along with the low level of student use, underscores the need for a high quality, general text book of computational molecular biology. Unfortunately, the different local flavors of computational molecular biology courses makes it difficult to define a core curriculum that should be shared, and could be the emphasis of a textbook.

We used electronic slides (made with a presentation package such as Microsoft PowerPoint™, projected through a flat panel display from a MacIntosh CPU) for approximately 70% of the course. The advantages of the electronic display included easy availability of lecture notes for students, and the ability to do real time demonstrations of software. One guest lecturer did an entire lecture using Netscape.

3. Conclusions

Our first experience teaching “Representations and Algorithms for Computational Molecular Biology,” offered as a course in Medical Information Sciences and Computer Science, was quite positive. The hands-on interaction with the data was appreciated by students, and has lead some to join computational molecular biology efforts. The short time available in one quarter, and the structural focus within bioinformatics lead to a syllabus that was somewhat incomplete in its coverage of current computational molecular biology. In the context of all the courses related to this area taught at Stanford, however, the emphasis on structural issues was not unreasonable and lead to a course that filled a niche. In addition, the constraints that biological structure places upon many aspects of sequence analysis make it useful to understand biological structure before focusing on sequence analysis. This runs counter to the standard view that the analogies between string matching and sequence analysis in computer science are so strong, that the provide the natural starting point for computational molecular biology courses. An understanding of structural principles often helps computer scientists avoid dangerous oversimplifications in sequence analysis. Therefore, we anticipate using the same strategy in the next version of the course, although we will place slightly more emphasis on sequence analysis. The principal modifications we anticipate are:

1. We will spend more time on the topic of dynamic programming. These algorithms are so fundamental to computational molecular biology, that an entire additional lecture should be spent on its variations, the different type of scoring matrices, and their theoretical underpinnings.

2. We will moderately reduce our emphasis on structural computation in the homework assignments. We will remove one of the assignments for structural analysis (for example, Assignments 3, 4 or 5) and instead have an assignment in which the students analyze the statistics of sequence and use these for classification.

3. We will provide a bit more biological background material for students. Despite the difficulty in adding extra material, we may have a more constant (as opposed to predominantly at the beginning) amount of biological reading, in order to better motivate the algorithms and methods that are presented. We will maintain the focus on a small number of biological systems (particularly the globins) in order to keep a manageable scope of required biological knowledge.

4. We will reduce the amount of readings, and try to integrate them into assignments more closely. It was clear that the large amount of programming impacted negatively on the ability of the students to read. The course reader was created to be a lasting reference for the students, containing some critical primary articles in the field. Although it is our preference that the students spend more time actively interacting with the data and programming than reading the primarily literature for the course, we are investigating ways to create a more readable syllabus of papers. There is clearly a need for a high quality textbook in this area, that covers the topics in our course, as well as numerous others.

4. Acknowledgments

RBA is a Culpeper Medical Scholar, and is supported by LM-05652. We would like to thank our teaching assistant, Simon Handley, for his excellent trouble shooting, and the course coordinator, Barbara Morgan, for her patience and flexibility, and the reviewers for their helpful suggestions. Some computing resources were provided by the CAMIS resource at Stanford, LM-05305.