Teaching Biologists to Compute using Data Visualization

Kay A. Robbins  
Department of Computer Science  
University of Texas at San Antonio  
One UTSA Circle, SA, TX 78249  
1-210-458-5543  
krobbins@cs.utsa.edu

David M. Senseman  
Department of Biology  
University of Texas at San Antonio  
One UTSA Circle, SA, TX 78249  
1-210-458-5543  
david.senseman@utsa.edu

P. Elizabeth Pate  
Interdisc. Learning and Teaching  
University of Texas at San Antonio  
One UTSA Circle, SA, TX 78249  
1-210-458-7442  
elizabeth.pate@utsa.edu

ABSTRACT
The accelerating use of computation in all aspects of science continues to widen the gap between student skills and expectations. Currently, computation is taught using one of two approaches: teach students a standard programming language (e.g., FORTRAN, JAVA or C) perhaps augmented by support tools such as Alice or teach them to use a program such as MATLAB by formulating and solving math problems. Both approaches have high failure rates for students hindered by poor mathematics training and weak logic skills. This paper describes an alternative approach that introduces students to computing in the context of data analysis and visualization using MATLAB. Our goal is produce computationally qualified young scientists by teaching a highly relevant computational curriculum early in their college career. The course, which integrates writing, problem-solving, statistics, visual analysis, simulation, and modeling, is designed to produce students with usable data analysis skills. The course is in its third year of implementation and is required of all biology majors at the University of Texas at San Antonio.

Categories and Subject Descriptors
K.3.2 [Computer and Information Science Education]: Curriculum, Computer science education

General Terms
Algorithms, Experimentation.

Keywords
Data Analysis, MATLAB, Visualization, Biology.

1. INTRODUCTION
In a recent editorial, Cooper and Cunningham [1] discussed the notion of teaching computer science in context and urged computer science educators to recognize the importance of "computational thinking" in a successful education [10]. They promote a broad view of computer science and encourage the development of computing capabilities in students outside of computer science as well as building a broad viewpoint for computer science students. ITiCSE'09 Working Group 6 [5]

defines the notion of "computationalist thinking" and articulates six concepts (modeling, abstraction, interpretation, scales and limits, simulation, and automation) that characterize and unite computational endeavors.

Providing a context for learning in the CS0-CS2 sequence underpins approaches such as Alice [2], media computation [4] and graphics-based curricula [7]. Studies have shown these approaches have been effective motivators and can provide good foundational programming skills.

Our own thinking on this issue evolved from the computational and statistical trends driving the field of biology as well as our experience managing a project that matched undergraduate biologists with quantitative partners (math or stat majors) to work on research teams conducting biological research. These students spent considerable time on data analysis activities for which they were singularly unprepared. In thinking about how to change the biology curriculum to make it more quantitative, we realized that computing was an important aspect of the transformation, but a single traditional computing course did little to remedy the deficiencies in the requisite student skills.

We envisioned an approach in which we taught computation by having students work on real data analysis problems. We selected MATLAB as a platform because of its high-level data analysis and user-interface support. However, most traditional MATLAB courses and textbooks emphasize formulas and target students with some mathematical training.

An exception is the Wright State EGR 101 which uses a combination of mathematics, computation and experiments to engage students in solving engineering problems at the freshman level [5]. Their course is noteworthy in several respects. They introduce MATLAB as an exploratory and computational tool to solve specific engineering problems that students will encounter in later courses. The beginning half of the course only uses high school mathematics, but the problems introduced are significant engineering problems. However, in contrast to the course proposed here, EGR 101 emphasizes analytical problem-solving rather than data analysis and does not have the teaching of programming skills as an objective.

The remainder of this paper has the following organization. Section 2 provides an overview of the curriculum, Section 3 describes course evolution, Section 4 discusses teaching issues, and Section 5 presents concluding remarks.
2. COURSE ORGANIZATION

CS 1173 Data Analysis and Visualization Using MATLAB is moving into its third year of implementation and is required of all biology majors at our university. The course has College Algebra as its prerequisite and students typically take the course in their sophomore or junior year, although advising encourages them to take the course as early as possible.

2.1 Topics

The course focuses around building skills in three areas: data analysis, computation, and scientific process. Specifically, we target the following skills.

Quantitative and data visualization skills:
- Display data using appropriate visualizations (e.g., histograms, pies, scatter graphs) and their combinations.
- Understand and use basic statistical indicators for data (e.g., mean, standard deviation, median, tests of significance).
- Make correct and relevant assertions about data.

Computational skills:
- Develop programs that use variables, relational expressions, loops, functions, and logical indexing.
- Apply basic functions (e.g., sum, difference, exponential, log) appropriately and in context.
- Manipulate data structures such as lists, tables, and strings.

Scientific skills:
- Formulate and test hypotheses about data.
- Use modeling to study scientific questions.
- Understand how to acquire and analyze data.

Figure 1 gives a week-by-week listing of the major topics with focus questions shown in italics. In the first five weeks, students learn to write scripts, call functions, and use the basic statistical indicators. We integrate rudimentary writing in the form of making correct assertions about the data throughout the course. After the midterm exam, we move to more complex programming skills needed in actual data analysis and to statistical analyses beyond mean, median, and correlation.

2.2 Course delivery

The course is offered in a 55-seat computer classroom, and each student has his/her own computer (currently multicore with 24” monitors running Windows XP). The room is equipped with iTALC classroom control software, which we find useful for locking screens and projecting demonstrations when the details on the main projectors are not visible to students sitting in the back of the room. We also use the ClassQue [8] software extensively for administering short exercises and reviewing student answers. In addition to the instructor, generally one or two assistants are in the classroom. We try to hire undergraduates with good interaction skills who have previously done well in the course.

1. Basic data visualization in MATLAB
   Introduction to the MATLAB workspace, basic plotting and editing of line graphs, loading files, saving plots, legends, figure captions, annotations, color, plottools
   How do I get started using MATLAB?

2. Effective use of line graphs, bar charts, and pies
   Line graphs, pie charts, bar charts, stacked bar charts, sum function, percentage calculations
   How can I display trends and relationships?

3. Combining plots to compare data sets
   Insets, multiple axes, units, bar charts maximum, minimum, mean, median, mode
   How can I combine graphs for effective comparisons?

4. Rates of change
   Plots of ratios, difference, per capita, slope and rate of change, rate of change per capita, fractions of the whole
   How can I characterize rates of change?

5. Scatter plots and logarithmic axes
   Scatter plots, curve-fitting, correlation, logarithmic axes
   How can I determine whether two variables are related?

6-7 Project and midterm

8. Vector indexing and program control
   Relational expressions, logical vectors, loops, if-else
   How can I extract information based on data values?

9. Depictions of spread
   Histograms, cumulative probability distributions, box plots, interquartile range, percentiles, standard deviation
   How can I depict the distribution of data values?

10. Depictions of error
    Error bars, standard error of the mean
    How can I depict uncertainty and variability in data?

11. Hypothesis testing and confidence intervals
    One sample and two sample t-tests, confidence intervals
    How can I determine whether an experimental group is significantly different from the control group?

12. Modeling
    Using difference equations and other models
    How can I simulate biological and other systems?

13. Real data (optional)
    Dates, low-level I/O, missing values
    How can I analyze real data?

14-15 Review

Figure 1: Week-by-week topic list

2.3 Course curriculum details

We have organized the course around lessons, labs, projects, and in-class activities. Our goal is to develop a hands-on course that includes significant problem solving as well as programming and writing. The course is designed to produce students with usable data analysis skills.
2.3.1 Lessons
We cover each major topic of Figure 1 in two lessons that introduce the material and give a core set of examples. We try to cover two lessons per week, which includes the students working through the lessons and participating in various in-class activities. The instructor gives some explanation at the beginning and during each lesson. Students can work through the lesson examples to see the process before fully grasping the details. Lessons provide models for how students should do their own computations.

2.2.2 Laboratories
The laboratories use the material from the lessons, but require students to apply the results of the lessons. Students can get help and are encouraged to collaborate on the laboratories. A lesson might have an example of using the mean function with an explanation of what it did and allow the student to try it out and observe the results. The follow-on laboratory asks students to compute the average for another data set and output it. A nearly identical statement would appear somewhere in the lessons (applied to different data).

2.2.3 Projects
Projects require extension and more application of material in an independent way. In a project, the student must decide what to do in addition to figuring out “how” to do it. The current version of the course has two projects: a graphical analysis of an existing data set and a statistical analysis of a more complex data set.

We typically use an epidemiological data set (such as diabetes or tuberculosis where the data is broken down by age group and by race) from the CDC or other government source for the data analysis project. Students may select any combination of graphs they choose to display the data, as long as they display all of the data and they choose combinations that enable them to make appropriate written observations about the data.

3. EVOLUTION OF COURSE MATERIALS

3.1 Lessons
We wrote the initial lessons while offering the course for the first time in the fall 2008 semester. In the initial version of the course, the laboratories and projects used real biological data, but most of the lessons used the non-biological traffic data set, count.dat, that comes with the MATLAB distribution.

Since its initial offering, we have revised the lessons several times as illustrated by two versions of an example from an early lesson shown in Figure 2. The top EXAMPLE 2, which came from the initial version of the course offered in the fall 2008, asks students to extract a variable to hold the first column of the count array. The example refers to the traffic data set, and the instructions describe the action in terms of arrays. The column corresponds to the traffic from intersection 1, as reflected by the variable name, but the example does not make the relationship explicit.

The bottom EXAMPLE 2, which comes from the current version of the course, uses a well-known biological data set [11], childhood diseases in New York City for the years 1931-1971. The example describes the variable in terms of its meaning (measles cases from 1931) rather than as a row or column of an array.

EXAMPLE 2: Separate array columns into variables
Type the following in the Command Window:

% Create variable for first column of count
inter1 = count(:, 1);

EXAMPLE 2: Define appropriate variables for analysis

% Measles cases in 1931 (row 1 of measles)
measles1931 = measles(1, :);

You should see a measles1931 variable in the Workspace Browser. The measles1931 variable holds a 12-element row vector containing the monthly measles cases for 1931.

In the space below:
Define a variable called mumps1942 that contains the monthly case counts of mumps for the year 1942.

Figure 2: Evolution of lesson examples. Top version is from first offering and bottom version is from the current offering.

After describing the expected results in terms of the data, the example asks the students to produce a similar MATLAB statement pertaining to a different row of a different array. As they work through this example, students learn to define variables (the assignment statement), to extract array rows and columns, and to associate meaning with rows and columns.

3.2 Use of FAQ-style questions
In a lecture-style course, instructors often present very detailed explanations of how things work and why. While this may work for an orderly and measured presentation of material, we have found that such a “waterfall” strategy does not work well for an immersion-style laboratory course. The second lesson already asks students to define variables, extract rows and columns from arrays, sum rows and columns of arrays, plot data using line graphs, and to make meaningful interpretation of data. We can present the details about how things work, but students are not ready to listen.

Instead, we have adopted the use of FAQ-style (Frequently Asked Questions) for delivering explanations as shown in Figure 3. During the class period students work through the lesson and we give a brief overview of the lesson objectives, ask and answer questions to test student understanding, and walk around and help students who are stuck.

After the lesson, students read over an associated questions page or FAQ that dissects and explains each statement in the lesson. Most of the important statements have many associated questions covering issues ranging from MATLAB syntax to the meaning of the commands. Our original format included the FAQ with the lesson in a single-column format with the answer following the question in italics. We now have a separate FAQ for each lesson that mirrors the examples and uses a two-column format with the question in bold on the left and the answer on the right. Students indicate that they go back to the FAQ many times.

3.2.4 Sleep diary case study
The second project has evolved into a major case study that we use throughout the course. We ask students to track their sleep...
patterns by keeping a 3-week sleep diary starting in the third week of the semester. Students record their bed times, their wake up times, whether they used caffeine that day and whether they woke up to an alarm and enter this information into a spreadsheet. This exercise exposes students to the vagaries of data collection. The entered data must meet rigid formatting guidelines, or the computer programs that use this data will crash. We grade this assignment as a gateway (full credit or nothing) and do not accept student data until it passes the verification tests.

We then consolidate, anonymize, and combine the data for all students (typically around 150 students in 3 or 4 sections). We email subject numbers to the students for use in comparing their particular sleep patterns to those of the class as a whole. We used this data for a number of examples in the lessons, a laboratory, and for the second project. We formulate questions such as "Are men's sleep patterns different than women's?" and discuss whether our cohort is representative of all people or of particular subsets.

In the final project, students compare their sleep diaries to those of the rest of the cohort. Students are free to select the statistics and graphs to compute within certain guidelines. They are also required to prepare a technical report evaluating the results of this analysis, reflecting on their sleep patterns, and discussing how these patterns and choices might influence their performance.

4. OTHER CONSIDERATIONS

4.1 Cell mode and scripting

MATLAB has a weakly dynamically typed scripting language that supports a variety of data structures. Most operations support arrays and other complex types, eliminating the need for loops in many of the usual data analysis situations. The power of the scripting language makes MATLAB scripts very compact but it requires some sophistication to use effectively. We spend a fair amount of class time having students work through simple examples by hand to make sure that they understand what is happening in the code.

MATLAB supports a mode of execution called Cell Mode. A MATLAB script is broken into cells by separating blocks of code by lines that start with `%%`. MATLAB can execute the code in any cell independently of the rest of the script using the current contents of the workspace. Users can break a script down into cells in any way they choose. Cell Mode encourages incremental development and testing. The lessons, which consist of 8 to 10 core examples, have each example in a separate cell so that we can execute each one independently of the rest of the script. Students can also examine the workspace variables and their values while executing the cells.

In the initial conception of the course, we envisioned using the MATLAB plottools and code generation to gently introduce students to scripting. We found however, that after a few hours of doing figures with plottools, students began to find the process very tedious. In later versions of the course, we have nearly eliminated the use of plottools. Instead we introduce MATLAB functions such as title, legend, and xlabel almost immediately. We then emphasize the importance of correct labeling of graphs rather than the visual details such as font selection. We did not anticipate that biology students with no programming experience would so easily adapt to scripting nor that they would so readily recognize the savings in time and effort such scripting affords.

4.2 Mathematics and statistics background

While the students in the biology program at our university are required to take a biocalculus course and a biostatistics course, this required MATLAB data analysis course only has college algebra as a prerequisite. While some of the students have completed their mathematics requirement, most have not. This presents challenges as well as opportunities.

We deal with 2D arrays from day one, but we stick to a spreadsheet metaphor during the course, referring to tables and lists rather than matrices. The analysis techniques covered in the course only require element-wise operations and are not difficult for students, once they master the notation.

We heavily rely on students knowing the equation of a line given two points. Students first meet this equation in the first week when we talk about what a line graph really is (lines connecting successive pairs of points). We meet the equation of a line again in the lessons on rates of change, where we use slopes and geometry rather than talking about derivatives. We emphasize the units of $x$ and $y$ when discussing rates of change. Lines are also an important element in the discussion of linear fits.

Many students are weak in their understanding of logarithms and exponentials. We do not approach this topic in a traditional manner. Rather, our discussion of logarithms emphasizes the use of log scales to view a collection of items that range in scale from small to large. We also talk about exponential growth in the context of graphs, discussing the fact that curves undergoing exponential growth appear linear on a semi log scale.

We do not assume much in the way of statistical background beyond knowing what the average is. We hand calculate means and medians as well as standard deviations. The course introduces the notion of a population and a sample in geometric terms.

The most difficult topic for students by far is the notion of hypothesis testing and the t-test. MATLAB has nice built-in functions for one-sample and two-sample t-tests. These functions have three return values: $h$ (an indicator of whether to reject the null hypothesis), a p-value, and a confidence interval for the mean or difference of the means. We apply the t-test in many different
situations and interpret the results without explaining the basis of the t-test in any detail.

As part of the statistical part of the course, we explain that the t-test assumes samples drawn at random from a normally distributed population. We discuss how well these assumptions are satisfied and apply visual and analytical tests for normality.

### 4.3 Role of hand calculations

We feel that pencil and paper calculations provide an essential check for students to make sure that they really understand what they are doing. Every day we ask students different types of review questions such as the following:

- **a.** Given the array \( A = \{1, 5, 2; 6, 2, 3\} \), what is \( \text{sum}(A) \)?

- **b.** The variable `horseTemp` represents a \( 24 \times 5 \) array containing the core body temperature in degrees Centigrade for five different horses over a 24-hour period. Write a MATLAB statement to define a variable `horse1` containing the temperature measurements of the first horse.

Question a. is a routine syntactical question that determines whether the student understands the MATLAB `sum` function. Students also have to understand the comma and semicolon notation for expressing arrays in order to answer the question.

Question b. determines whether a student understands how to extract a column from an array and how to use an assignment statement. The question has extraneous information and requires more than just syntactical understanding.

We also give a few hand-calculation homework problems that we check off as completion grades and go over in class. For example, we ask students questions about the table at the left such as:

- **a.** Find the maximum daily count for Beach 1.

- **b.** Find the fraction of Beach 2’s count that occurred on Day 6.

- **c.** Find the percentage of Day 5’s count that occurred on Beach 1.

<table>
<thead>
<tr>
<th>Day</th>
<th>Beach 1 CFU/100ml</th>
<th>Beach 2 CFU/100ml</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>80</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>50</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>10</td>
<td>50</td>
<td>30</td>
</tr>
</tbody>
</table>

After going over the answers, we write MATLAB code to calculate the quantities.

### 4.4 Modeling

We incorporate modeling in several ways. We introduce linear models in the lesson on linear fits, scatter plots, and correlation. We do several examples using linear fits to make predictions about the data. For example, one of the labs looks at cricket chirps versus temperature to assess the accuracy of a "cricket thermometer", a well-known backyard science project. We also read the famous Nature paper by Dawson and Reid [3] that uses linear fits to derive an equivalence between alcohol consumption and sleep deprivation.

During the statistics unit, we perform simulation experiments to measure how closely the mean or standard deviation of a sample reflects the mean or standard deviation of the underlying population. We sometimes do simulations using nonlinear difference equations and reproduce the bifurcation diagrams.

### 4.5 Writing

Requiring students to articulate their understanding through writing is an important tool for developing higher-level data analysis skills and for assessing student understanding of these concepts. As the course has evolved, we have incorporated a writing component that is manageable for both students and instructors. We define the following levels of writing:

- **Level 1 (mechanics)** includes grammar, spelling and sentence construction that can be addressed to a significant extent by spelling checking and grammar checking tools.

- **Level 2 (facts)** includes correct and informative interpretation of the data.

- **Level 3 (insight)** involves making connections among multiple pieces of information and describing relationships in a broader context.

Our goal is for students to write consistently at Level 2 with a smidgeon of Level 3. We require students to use Microsoft Word with grammar checking turned on and forbid them to use the word “it”. We can check these requirements mechanically and find they are sufficient for students to meet the Level 1 requirements. We require students to submit major writing assignments through Safe Assign on Blackboard to eliminate plagiarism.

In the initial versions of the course, we required students to write half page reports in conjunction with various labs and assignments. We found that poor writing confounded our ability to pick out logic errors. Our graduate student graders, many of whom were non-native English speakers could not deal with the grading. We now do most of the writing (except for the final project reports) in the form of bullet points. TAs cut and paste incorrect bullet points into the grading program comment box and assemble all of the incorrect answers in a single file to give to the instructors. They do not offer corrections. The instructors can use the file as the basis for in-class writing exercises.

An important insight that we have gained while working through our writing strategy is that it is relatively easy to demonstrate bad writing—students provide a rich example set. However, we have found it essential to model good writing before we ask them to do their own writing. In particular, instructors must give many examples of good observations about data in a variety of contexts in order for students to be successful.

### 4.6 Classroom management

Keeping students on task in a computer classroom is always a challenge. Installation of the ITALC classroom control software helped a lot. Pacing is everything, and we have found that it is important to have activities available for the early finishers (or get them to help the other students). Attempted use of distracters went way down after we put a strong statement in the syllabus about classroom courtesy and stated cell phones and other devices used during class were subject to confiscation.
5. DISCUSSION

This paper describes the design, implementation, and evolution of a data-driven computing curriculum for undergraduate biologists. The project is well into the implementation phase, and we have offered 11 sections of the course taught by six different instructors. We are entering into the summative evaluation phase of the project. We have developed a complete set of teaching materials including lessons, lecture plans, laboratories, projects, and teaching notes, all of which are available on the web at http://www.cs.utsa.edu/~cs1173. The course materials have undergone a number of rewrites as we have learned what works and what does not. The course includes a writing component and discussion of actual scientific literature. The remainder of this paper is a summarization in the form of a FAQ.

Can this course be done without MATLAB? Ans: One of the problems with traditional programming courses and languages is that you have to work very hard to do anything useful. It takes most of the semester in a beginning course for students to reach the point of being able to read values from a file into an array in a typical beginning Java, C, or C++ course. Graphing takes an extraordinary amount of work. Students can do little computation that is useful to a working biology lab after taking a traditional programming course. Our students have useful data analysis skills after completing this course. Our students learn a lot about computers, programming, and debugging along the way. Other environments such as SciLab, Octave, or SciPy or even Excel [9] could be adapted to integrate programming and data analysis. We do rely on the strong debugging and visual data browsing support provided by the MATLAB IDE that MATLAB substitutes don't always have.

Can this course be done without a computer classroom? Ans: We don't think so. There are some attractive possibilities with having two students share a computer instead of having each student at his or her own computer. However, the course material requires students to have extensive hands-on experience with someone readily available to help when they become stuck or confused.

Does this course have to be about biology? Ans: No. We could easily substitute data sets from the other sciences, engineering, psychology, and economics for those used in the current examples to change the orientation of the course. We choose the particular data analysis techniques to reflect common usage in biological papers. However, we do not have any examples related to genetics or bioinformatics in the current version of the course. A biologist is an expert in drawing conclusions from data analysis. With the right kind of data analysis and visualizations, students can come up with any number of research questions that they would like to work on. They can then work through the lessons independently. Their difficulty comes from trying to adapt the code to slightly new situations. A hybrid course would require a collection of video demonstrations and probably additional online material.

Could this course be offered as a hybrid course? Ans: Maybe. While we do not think that a course offered entirely online is feasible, some combination of computer lab and online course may be possible. After the first few weeks, students are able to work through the lessons independently. Their difficulty comes from trying to adapt the code to slightly new situations. A hybrid course would require a collection of video demonstrations and probably additional online material.

Can we make connections to other courses? Ans: We are trying. The statisticians are planning to make this course a corequisite for the required biostatistics course so that students will have a common computing background. When we first start introducing some of the statistical concepts such as the concept of normality and the t-test, students without statistics feel they should have statistics first. However, we have found that having the statistical background really does not appear to have a strong influence on how well students do. They appear to benefit from acquiring a more geometric grasp of the concepts before listening to the mathematical explanations from their statistics course.

This course is a prerequisite for a required junior level course in physiology. Work has begun on redesigning the curriculum for that course and we hope to modify some of the examples to be more relevant to physiology in nature. In particular, we plan to develop some examples using a model of the heart that students see in biocalculus and use for some analysis units in physiology.

6. ACKNOWLEDGMENTS

This work was supported by NSF grant 0837248.

7. REFERENCES


