DEVELOPING INTRODUCTORY COMPUTING FOR STATS UNDERGRADUATES

Paul Murrell
The University of Auckland, New Zealand
paul@stat.auckland.ac.nz

Computer literacy is an essential part of a modern education in statistics (or any field that involves working with data). Two statistical computing courses have been developed in the Department of Statistics at the University of Auckland. The first provides instruction in fundamentals of computer storage, writing computer code, and data structures. The second covers simple programming ideas, how to write larger pieces of code, and graphics. This paper describes the evolution of these courses, noting successes and failures both in terms of what content is delivered and in terms of how that content is delivered.

INTRODUCTION

In 2001, the Department of Statistics at the University of Auckland began to develop a statistical computing stream for undergraduate statistics students. The primary motivation for this development was the realization that computing skills were becoming increasingly important in both statistical research and statistical practice, but that the department's curriculum was not providing sufficient instruction in statistical computing. Another goal was to develop graduate students with the necessary computing skills to be able to conduct research projects in statistical computing.

The first course to be developed was called "Data Technologies". This is a Stage 2 (second-year) course with a broad focus on computing technologies that relate to working with data. More recently, a Stage 3 course called "Statistical Computing" has been developed, to focus more on computer programming, particularly with the R language and environment for statistical computing and graphics.

This paper describes the evolution of these courses to date, noting lessons learned from various successes and failures along the way.

DATA TECHNOLOGIES PART I

The original idea behind the course "Data Technologies" was to provide an introduction to a variety of computer technologies that are relevant to working with data. The overall structure of the course was initially designed around the "lifecycle" of a data project:

- HTML Forms as a technology that facilitates *data collection*;
- XML and Relational Databases as technologies that facilitate *data storage*;
- SQL as an example of a *data retrieval* or *data access* technology;
- PHP as a technology for *data processing* (this included an introduction to Regular Expressions).

In addition to providing an introduction to these specific technologies, a goal of the course was to nudge the students away from being just *users* of computers (a little way) towards being computer programmers.

The course was delivered in the form of two Lectures and one practical Lab Session (with students working on computers) each week. Assessment consisted of the lab work and three Assignments, plus a Term Test and an Exam (the latter two written by hand without the benefit of any computer to offset the ease with which assignment and lab work could be "reproduced").

A secondary goal in the initial development of "Data Technologies" was to create a course that could be delivered online. To this end, all lecture slides and lab and assignment handouts were made available on a course web site. Students were also able to submit labs and assignments online. Furthermore, the technologies that were not already natively supported by a standard web browser (SQL, Relational Databases, and PHP) were provided via online services; the course server ran a MySQL database server, to which students could submit SQL queries via web forms, and an Apache web server, which had the PHP module enabled so that students could upload and

In C. Reading (Ed.), *Data and context in statistics education: Towards an evidence-based society. Proceedings of the Eighth International Conference on Teaching Statistics (ICOTS8, July, 2010), Ljubljana, Slovenia*. Voorburg, The Netherlands: International Statistical Institute. www.stat.auckland.ac.nz/~iase/publications.php

run web pages containing PHP code. All of this meant that the students only required a text editor and a web browser in order to complete all lab and assignment work for the course.

Successes

The provision of online course materials has now become standard practice; it clearly allows students to consume material in a variety of ways and at their own pace. Student feedback is only positive about this facility. If success can be measured by how many students attend lectures, then the provision of online material for "Data Technologies", with an attendance rate of about or below 50%, can be considered a success.

Electronic submission of coursework succeeds in one way because it uses much less paper. More importantly, it is crucial when students are submitting computer code for assessment because it allows the computer code to be *run*. As well as allowing the code to be properly tested, an important consequence is that the marking of submissions can be at least partly automated. Another important feature about the submission system was that a copy was sent back to the student so that they could feel confident that their work (and the correct *version* thereof) had been recorded.

Failures

The choice of PHP as a data processing technology was misguided. PHP did allow a nice "closure" for the course because the students were able to end up creating fully-functioning web forms with HTML Form front-ends and PHP back-ends. However, PHP is not a technology that is typically employed for data analysis. This error was compounded by the manner in which this section of the course was delivered. There was an attempt to introduce basic programming concepts, such as loops, conditional statements, which were too much too soon for many students, particularly because the course failed to provide any good reasons for needing these programming concepts.

Unfortunately, without PHP (or something similar, such as javascript), it is not possible to adequately address HTML Forms. It is much less useful to teach students to create forms if those forms do not actually "do" anything.

DATA TECHNOLOGIES PART II

The current format of the course "Data Technologies" has shifted away from the focus on a data project lifecycle, to a focus on taking control of the computing environment. This focus still requires learning a number of technologies, which all have a connection to working with data, but it requires spending more time on understanding simpler and more fundamental computing concepts. The course now consists of the following topics:

- HTML as a simple example of a *computer language*;
- Computer Memory and File formats as fundamental knowledge about *computer storage*;
- XML and Relational Databases as technologies that facilitate *data storage*;
- SQL as an example of a *data retrieval* or *data access* technology;
- R as a technology for *data processing* (this includes an introduction to Regular Expressions).

HTML is used to introduce basic ideas of writing computer code; for many students, the idea of communicating with the computer via the keyboard rather than the mouse is very alien. In this part of the course, we address simple ideas such as indenting code and writing comments. The section on data storage formats is similarly focused on challenging the students' view of the computer. Many students struggle with separating files (documents) from programs (applications) and are surprised at being able to open a web page using a text editor rather than a web browser.

R replaces PHP for data processing, which makes much more sense as a language for working with data. However, just as significant is the change in this section of the course away from abstract programming concepts. Instead, there is a much larger emphasis on *data structures*. This still provides a lead-in to some simple programming ideas, but from a much more fundamental starting point. Furthermore, the material is presented from a problem-solving point-of-view: we want to perform a data processing task *X*, which means that we need to know *Y*.

The assessment format of labs and assignments plus test and exam has remained exactly the same.

Successes

The change in emphasis to more fundamental computer concepts is much more aligned with the knowledge and ability level of the students. Student evaluations have improved over time.

The change in technologies (R to PHP) has made the course more relevant to a Data Technologies course (rather than a Web Technologies course).

Failures

Removing HTML Forms has removed a lot of the reason for HTML, although fortunately HTML is sufficiently ubiquitous for it to still remain relevant. Removing HTML Forms has also removed some of the overall coherence of the course; for example, the end does not tie as neatly back to the beginning.

STATISTICAL COMPUTING

The second course to be developed was called "Statistical Computing". This course is focused entirely on the R language and environment for statistical computing and graphics and has much more of a focus on teaching the students how to *develop* new code, rather than just calling existing functions. In just four years, it has already had a quite turbulent history.

The first incarnation of the course involved three instructors, who split the course into three components: programming (functions, control flow, and object-orientation), computational (numerical analysis and simulation), and graphics. This was not a huge success. The programming section was too ambitious, the computational section too theoretical, and the graphics section too abstract.

A second iteration involved only two of the instructors, with the same set of topics, but this enjoyed no greater success (neither in terms of student evaluation nor instructor satisfaction).

In 2009, the set of topics was reduced, with most theoretical content and any mention of object-orientation dropped from the course. Furthermore, one half of the course was based on a single, large case study that involved harvesting a set of data from a set of web pages. The purpose of using the case study was to provide a clear motivation for more complex data structures and data manipulation functions in R. In addition, the larger problem involved writing a larger set of R code, which lead to a discussion of strategies for building up, maintaining, and managing larger blocks of code. The emphasis was not so much on becoming a programmer as on being able to write larger pieces of code to perform more complex and ambitious tasks. For example, writing a function can be introduced as a way to make code tidier and easier to maintain. This also provides many opportunities to emphasize the importance of breaking a larger task into smaller pieces. The class is also repeatedly told to write code in small pieces, test those pieces on simple (made-up) examples, and then build more complex tasks by combining small pieces of code that are known to work.

Successes

The students appreciated being able to see a reason for learning the R functions and being able to see how the functions could be used together to produce more complex results.

> "I really enjoyed the course and especially liked how you taught the second half with the 'big project' theme - it made following what was going on much easier as we could see exactly how we were progressing." *(spontaneous student feedback)*

Failures

While a case study makes it easy to motivate learning certain R functions, it does not provide an opportunity to cover *all* possible uses of a function. The balance needs to shift back towards the abstract teaching of facts in small sections in and around the overall case study so that students have more information to help them adapt the lessons from the specific case study example to other data sets and other data processing tasks.

CONCLUSION

The approach to teaching statistical computing that has been most successful is one that begins with the data. Virtually any data analysis project involves a variety of problems relating to data storage and data processing. This makes it very simple to motivate the need for computer tools.

In the Statistics Department at the University of Auckland at least, many of the students have very poor computer literacy, so it is necessary to address quite fundamental computer concepts before moving on to teaching useful computer technologies.

Most of the students will not end up as programmers, but they will end up writing code. At least as important as teaching the computer technologies themselves is the need to teach students how to write tidy code and how to construct larger pieces of code from small pieces. A solid grounding in fundamental data structures is also essential.

REFERENCES

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (2008). *Extensible Markup Language (XML) 1.0 (5th Edition)*. W3C. http://www.w3.org/TR/2008/REC-xml-20081126/.

Friedl, J. E. F. (2006). *Mastering Regular Expressions (3rd Edition)*. Sebastopol, CA: O'Reilly.

Groff, J. R, & Weinberg, P. N. (2002). *SQL: The Complete Reference (2nd Edition)*. Berkeley, CA: McGraw-Hill.

Hoffer, J. A., Prescott, M. B., & Topi, H. (2009). *Modern Database Management (9th Edition)*. Prentice Hall.

Murrell, P. (2009). *Introduction to Data Technologies*. Boca Raton, FL: Chapman Hall.

R Development Core Team (2009). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Online: www.R-project.org.

Raggett, D., Le Hors, A., Jacobs, I. (1999). *W3C HTML 4.01 Specification*. W3C. Online: www.w3.org/TR/1999/REC-html401-19991224/.