Vistas in Advanced Computing

Amit Amritkar*, Jerry Ebalunode†, Martin Huarte-Espinosa‡,
Peggy Lindner§, Pablo Guillén Rondón¶ and Andrea Prosperetti**‖
Center for Advanced Computing and Data Science, University of Houston
Houston, TX
**Also: Department of Mechanical Engineering, University of Houston
Email: *aramritk@central.uh.edu, †jebaluno@central.uh.edu, ‡mhuartee@central.uh.edu,
§plindner@central.uh.edu, ¶pgrondon@central.uh.edu, ‖aprosper@central.uh.edt

***Abstract*—This paper describes a summer program, "Vistas in Advanced Computing," in which rising sophomores were instructed on major components of High Performance Computing including C programming, numerical methods, computer architecture, parallel programming and other topics over a period of eight weeks in the summer of 2017. Students spent 6-7 hours a day in a classroom attending lectures, studying, doing homework and coding. The students were very positive about the experience, an evaluation which is supported by the data collected in the course of the program.**

***Index Terms*—Undergradaute training, HPC, Scientific Computing**

I. Introduction

The ever-increasing role of computation in modern society requires novel approaches to the teaching of many of the relevant skills. This need is motivated not only by considerations of efficiency but also – and perhaps more importantly – by the need to expose STEM students to meaningful applications of computers as early as possible in their career. Failure to make their education feel more relevant for the world in which they live runs the risk of causing many to abandon the field, which is a negative outcome for society and for their personal life trajectories as well. While nearly 40 percent of the students entering 4-year post-secondary institutions indicate a plan to major in STEM fields, about half of them fail to achieve this goal (Malcom & Feder, 2016). Certainly many factors lead to this outcome, but the distance between what students are taught in their introductory math and science classes and what they perceive as scientific and technological problems relevant to the world that surrounds them may be large enough to induce a sort of disaffection toward their chosen field.

In this paper we describe an experiment in which we have spent 8 weeks in the summer of 2017 teaching 9 rising sophomores advanced computing, numerical methods, computer architecture and parallel computing, with exposure to other topics such data mining, machine learning and molecular dynamics. Given the bredth of topics it was impossible to go to much depth in any of them, which was the motivation to title the program "Vistas in Advanced Computing." Our plan is to continue to involve this cadre of students in research projects in the summer of 2018 and to find suitable internships for them for summer of 2019. We chose to hold the program during the summer months to avoid lengthening the time to graduation, which has become a matter of serious concern in the past two decades (see e.g. Bound *et al.*, 2010; Yue & Fu, 2017), catching the attention of policy makers and educators as well as the general public.

Our program differed in two significant ways from typical semester-long courses in HPC. In the first place, our students had just completed their freshman year and had, therefore, a much more limited background than the typical audience of such courses. As a matter of fact, we believe that demonstrating that it is possible to make the material accessible to students so early in their career is one of the intersting results of our experiment. Secondly, our program encompassed an array of ideas and skills broader than what is usually offered in HPC courses. Several national labs offer HPC bootcamps, typically running over a single week and, therefore, much shorter than our program and mostly directed, again, to a more mature audience.

Our students spent about 7 hours a day in a classroom setting, part of the time attending lectures given by us and part of the time studying, solving homework problems and coding. All had completed the standard Calculus sequence and introductory courses in Physics and Chemistry, but they had not taken courses in differential equations or linear algebra yet. It should be stressed that, while these students were all from honors classes in engineering, mathematics, physics and computer science with a GPA of 3.6, we made no effort to identify particularly gifted subjects for our experiment. For reasons of time we were only able to advertise the program to a limited extent and we accepted the students who applied. Our only screening was at the high end, rejecting an applicant who was too advanced to be a good fit for our program.

We are very happy with the results of this experiment. Students were taught material that undergraduates would encounter in the course of their normal studies only much later, if ever. Their response was unanimously enthusiastic and led some to recalibrate their career plans, with a stronger orientation toward research. Additionally, mathematics came to life for them in a way that they had never experienced. On the basis of this experience, we feel that a focus on computation in the very early years of the college STEM education may also be beneficial in fostering and motivating the learning of mathematics, which is a notorious sore point in the US education. It is well known that the United States

is a country with a larger proportion of low performers in mathematics than the OECD average, a situation that has not improved since at least 2003 (PISA, 2016).

The purpose of this paper is to describe the structure of our program, the topics covered and to present an assessment of its outcomes.

II. Advertisement and retention

As already noted, due to reasons of time we were unable to broadly advertise the program. One of us simply asked instructors of several honor Calculus and Physics classes for a few minutes at the beginning of a class period and briefly explained the program to the students, who were directed to a web page for further information. While the number of students we were able to reach in this way was limited, it must be recognized that the program that we have developed is not suitable for a significant expansion without a major increase in resources. We were able to accommodate 9 students and, for reasons that will become clear in the following, probably twice as many represents the upper limit at which our program can maintain its effectiveness. This is our target number for the summer of 2018 when the same program will be offered again.

Since many students work during the summer, we felt it was necessary to offer a financial incentive to make participation feasible for our students. Each one of them received a scholarship of $ 4000 for the 8 weeks of the program, supported by in-house funds. Doubling the number of participants will increase the cost and we plan to look for additional support from philanthropic and governmental sources.

We were very happy to find that retention was not a problem in the least. The students quickly developed a strong *esprit de corps*, felt privileged to be part of the program and realized the uniqueness of the experience they were offered.

III. Overview

The program was articulated in four main themes, described in greater detail in the sections that follow:

1) The C programming language;
2) Basic numerical methods for ordinary and partial differential equations and linear algebraic systems;
3) Computer architecture;
4) Parallel programming with OpenMP and MPI.

Two other topics, machine learning and molecular dynamics, were covered more briefly. In addition, there were weekly seminars addressing such topics as chaotic phenomena, plasma ejection from stars, the Leidenfrost phenomenon and how to make the most from attending a conference. All the authors of this paper shared the development and presentation of the material and a TA was available to help students with their assignments.

The last week of the program was devoted to individual mini-projects consisting of the study of a specific topic, the programming of the relevant mathematical model, the analysis of typical results, the presentation of the work to the class and the preparation of a written report. Students were encouraged to drive the project in the direction they found most interesting, a freedom which helped them develop their understanding of the subject. The topics were in the following areas:

1) Traffic flow;
2) Molecular dynamics;
3) Lotka-Volterra model of two-species dynamics of biological systems;
4) The Hodgkin-Huxley model of action potentials in neurons;
5) A simple model of star formation;
6) The Burgers equation.

Students were asked to organize their projects (codes, reports, test cases) in different directories to convey to them the concept of software packaging and redistribution. They were also asked to describe the system on which they had executed their projects to give them a sense of hardware influence on the performance of their codes.

The plan for the second summer of the program (2018) for the cohort of students who have completed the first summer is to assign to each one more substantial research projects, developed with the aid of colleagues in the UH Colleges of Engineering and Natural Sciences and Mathematics, to be carried out over the time span of one-two months. For the third summer we will help the students to find internship in UH laboratories and area companies.

In order not to lengthen time to graduation, we felt that it was not appropriate to involve the students in intensive activities during the regular fall and spring semesters of the academic year. We plan to bring them together again approximately once a month for lectures, conversation and socials.

IV. The C programming language

The first two weeks of the program were devoted to intensive instruction on C programming. No student had had any previous exposure to this language. After a short introduction on the history and evolution of C and its relationship to other existing languages, the course turned to basic programming: the idea of the main function, user-defined functions, memory allocation, variable declarations and, eventually, control instructions. Every morning students were shown YouTube videos covering the material taught during the previous day with the aim of reinforcing their knowledge, both by the repeated exposure to the material and by hearing it from a different perspective from that of the instructor. The class then proceeded to cover the day's new material and address questions. During the afternoon, students were given a set of about 10 problems and 1-1.5 hours to solve them. Next, the instructor randomly selected students each student, in random order, to explain to the class how they had solved the problems and their reasoning in writing the code. A great deal of discussion and positive feedback was encouraged. Students very quickly seemed to develop a team spirit, with everyone involved in helping the others.

By the beginning of the second week, students showed proficiency in reading programs, writing some simple ones and,

also, pointing out alternative algorithmic implementations. Advanced topics, such as pointers, structs, data structures, and I/O were then covered. Students were excited, confused, and eventually satisfied with their new abilities to understand programming paradigms, and being able to code some of these very powerful features of the C language. These classes ended with students choosing one of two final project options: make a Sudoku solver, or a Morse Code translator. They worked in teams of two, and were given 5 days to finish. All results were timely and interesting.

Another component of the material taught to these students dealt with Data Visualization. Fundamental concepts were discussed with the intent of cultivating best practices. Since most of the students were already familiar with MATLAB, data visualization using MATLAB was covered first. Then the coverage moved to the use of high-performance tools, VISIT and PARAVIEW. The material was designed with STEM applications in mind, making use of sample data relevant to engineering, physics and medicine, all of which are topics of interest in the Houston area job market. Amongst other exercises, students learned how to visualize and analyze tumors embedded in human brain CT scan, how to analyze turbulent flow moving behind airplane wings, how to create animations of vector fields from static data. They were shown how to create high-quality plots, graphics and animations, worthy of scientific publications or conference presentations. There was a lot of discussion on how to utilize this kind of tools for other computer applications as well.

V. FUNDAMENTALS OF NUMERICAL METHODS

Among the many possibilities in this area we decided to focus on methods in the general area of scientific computing. This choice was motivated by the desire to help students realize the essential connection between the mathematics taught in Calculus classes, the mathematical form of the natural laws taught in Physics classes and the remarkable power of computation in generating results beyond the elementary ones to which they had been exposed. Lectures ran for about two hours a day for two weeks. The topics covered were:

- Root finding;
- Numerical approximation of derivatives, low- and high-accuracy formulae, truncation error;
- Methods for ordinary differential equations and systems: explicit and implicit first-order methods, predictor-corrector methods, Runge-Kutta methods, adaptive integration;
- One-dimensional Laplace, Poisson and Helmholtz equations: Dirichlet and Neumann boundary conditions, ghost nodes;
- Linear algebraic systems: the tri-diagonal algorithm, Jacobi iterations, Gauss-Seidel iterations;
- The diffusion equation: explicit and implicit methods, Crank-Nicolson method.

After each class the students spent the remaining 5-6 hours of the day writing code to solve problems by the methods taught in the lectures. They found the material challenging at first, but quickly caught up with it, became intrigued by the applications and were successful in the numerical implementations.

VI. COMPUTER ARCHITECTURE

In addition to the fundamentals of mathematics and coding, it is imperative that students acquire some knowledge of the computational hardware. By better understanding the process by which their codes get executed, they are then able to make them more efficient. This material was covered over two weeks, in parallel with the numerical methods component, mostly in the form of lectures and hands-on exercises. Student progress was evaluated by means of homework assignments and pop quizzes. The focus was on the following basic concepts:

- Numbers and computers: bits/Bytes, reality of numbers, precision/significance and errors;
- Computer hardware trends;
- Overview of hardware components in an actual node: identify different components visually; identify hardware configuration from command line;
- Code compilation and execution process and how it maps onto the hardware;
- The Memory Hierarchy: Disk drives, RAM and Caches; locality for CPU-memory gap and how to program for it; cache analysis and writing cache friendly codes;
- Instruction level parallelism and vectorization.

VII. PARALLEL PROGRAMMING

After a brief review of parallel computer architectures and characteristics of different network topologies, students were led to conceptualize the idea of performance in parallel applications by the introduction of parallel metrics. Based on Flynn's taxonomy for parallel computer architectures, various parallel programming paradigms were introduced, focusing mostly on Message Passing Interface (MPI) and shared memory programming (OpenMP) paradigms, along with a short discussion about POSIX threads.

The course material was covered in one week with many hands-on practical exercises that focused on the design and analysis of parallel algorithms. In addition to daily homework, the students had to write a response paper comparing two journal articles that discussed the development of multi-core processors Gepner & Kowalik (2006); Gepner *et al.* (2007).

For the MPI part the students were first introduced to distributed memory computer architectures and the general idea of the message passing model using point-to-point operations (blocking/non-blocking). We then covered collective operations and students started to work through simple examples derived from scientific computing, including solving systems of linear equations and matrix operations. We also touched upon the topic of debugging parallel applications.

The OpenMP part involved writing parallel code by implementing OpenMP directives. Topics covered included the fork & join execution model, many-core processor computer architecture, data scoping, work-sharing, reductions, synchronization, and use of OpenMP functions. Students modified several

sequential codes to run in parallel, benchmarked their performance, identified performance bottlenecks for bandwidth-starved applications running on NUMA platforms and implemented work-arounds, such as first-touch memory allocation, to improve the application performance.

VIII. Additional topics: Machine learning, molecular dynamics and others

In addition to the series of lectures covering the topics described above, we felt that it was useful to give the students a very cursory exposure to topics of great current importance and also likely to pique their curiosity. For this purpose two days were devoted to machine learning and molecular dynamics. In keeping with the "hands on" philosophy of our program, for both topics there were lectures for about half the time with the rest of the time spent on exercises and homework problems. Furthermore, we briefly covered a few topics falling under the heading of "good computing practices."

MACHINE LEARNING. The starting point was the concept that computers can be programmed so as to learn. The students were shown how to "talk" to a computer in a way that is simple, yet powerful and, most importantly, in a way that can be readily understood by it. The basics of data mining and machine learning were outlined, discussing concepts like linear regression, classification, model evaluation metrics, and model selection. Students had access to a Linux cluster to run several machine learning algorithms as implemented in powerful packages such as SCIKIT-LEARN (Pedregosa *et al.*, 2011) and H2O. In this way they gained an understanding of core concepts in machine learning and began to master the ability to efficiently train and benchmark accurate predictive models. The topics covered included

- Data mining, pattern recognition, and approaches to extracting features for classification tasks;
- Unsupervised and supervised learning;
- Introduction to hierarchical clustering;
- Splitting the available data for training and testing purposes in machine learning;
- Implementing the SCIKIT-LEARN classifiers: logistic regression, linear discriminant analysis, k-nearest neighbors, support vector machine, evaluation of the accuracy of the models;
- Introduction to deep learning with $H2O$: designing architectures for specific tasks, and evaluating the accuracy of the models.

After the theoretical exposition of each topic, students spent some time running the algorithms and learning to fit and evaluate models using the Python tools mentioned above, with examples drawn from Neuroscience Guillén *et al.* (2011, 2012).

MOLECULAR DYNAMICS. The Vistas scholars were introduced to the principles of molecular mechanics and molecular dynamics as applied to biological systems. The theory behind the MD method was briefly covered followed by a rapid introduction to carrying out molecular dynamics on biological molecules of interest such as proteins and enzymes using the state-of-the-art Python programming OPENMM molecular dynamics application toolkit. Students were able to quickly develop their own custom MD simulation applications targeting different ensembles, and to emulate experimental conditions in a wet lab setting. Special attention was given to describing the workflow from the point of getting the starting coordinates from the protein data bank database, structure preparation, MD simulation and analysis of the resulting trajectory. Several of the students went further to use MD simulations to characterize the binding energetics for protein-protein complexes involved in cancer pathogenesis. These applications illustrated how the general goal of developing new strategies for the design of anti-cancer drug agents can be addressed by MD methods.

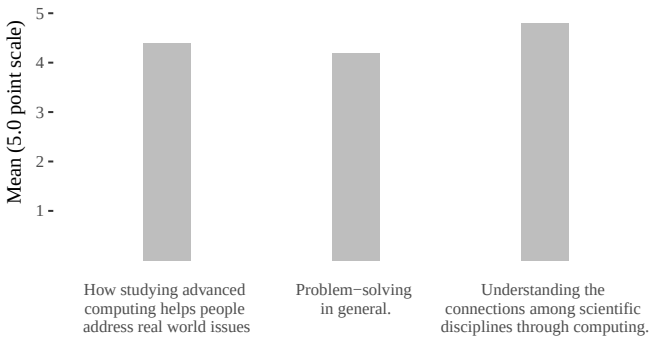

Fig. 1. Students' gains in understanding

GOOD COMPUTING PRACTICES. Since the students were mostly new to programming, it was desirable to arm them with the tools necessary for success. Debugging is an activity which is immensely helped by the use of correct tools. So the students were taught the use of appropriate compiler flags and GNU debugger (GDB/DDD). The use of the debugger was illustrated in class with extensive hands-on exercises. Some students did use these tools for their subsequent coding assignments and projects.

Version control is also an important tool to organize and manage documents and codes which is normally not covered in computing courses. Our students were taught to carry out their projects using `git` for version control. This also allowed us to streamline the process of assignment submission and evaluation by asking the students to use GITHUB or BITBUCKET accounts.

IX. Student evaluation

The various quizzes, problem sets and projects daily assigned by each instructor provided us with a way of continuously evaluating the students' understanding and progress and, at the same time, of gauging the effectiveness of our approach to teaching the various topics.

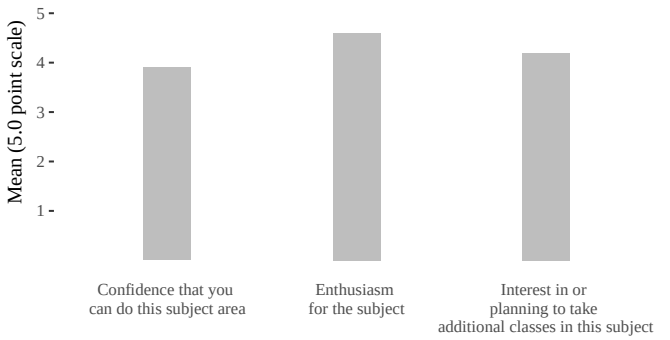

Fig. 2. Students' gains in attitude

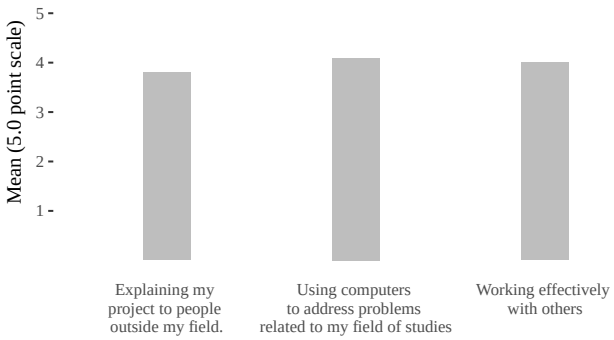

Fig. 3. Students' gains in skills

At the end of the program we administered electronically the Student Assessment of Learning Gains (SALG) survey (Seymour *et al.*, 2000) to carry out a systematic evaluation of the students' confidence and interest in scientific computing, as well as of their perceptions about their summer's learning experience.

The understanding, confidence and enthusiasm towards scientific computing showed the highest gains. The students thought that the learning experience and classroom instructional environment helped their learning. They rated their interactions with peers and instructors positively and also reported that specific class activities, such as the hands-on exercises and participation in discussions, were helpful.

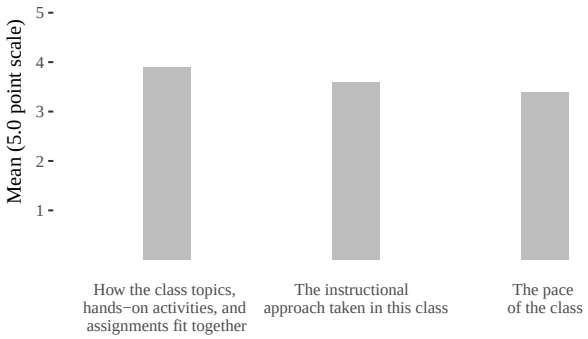

Fig. 4. Students' learning experience

A. Student interest

Figure 1 provides a break-down of student responses with respect to their understanding of scientific computing. They all reported that the program had significantly increased their understanding of "the connection among scientific disciplines through computing". With one exception, they said that, even though they already had an interest in scientific computing before, the program had increased their "enthusiasm for the subject" a "good" or "great" amount and planned to take more scientific-computing-related courses in the future (figure 2). In answering an open-ended question, the vast majority of the students affirmed that the program greatly increased their personal interest in the field and they had changed their attitudes towards the willingness to learn beyond the requirements of their degree.

Most students expanded their answers noting that they had gained a greater depth of knowledge about the topic. The vast majority (8 out of 9) reported a significant increase in their skills, especially for "using computers to address problems in their field of study" (figure 3). Two thirds said that the program had significantly increased ("good" or "great" gain) their confidence in their ability to succeed in scientific computing, while 3 reported only "moderate" gains.

B. Learning experience

Students rated the learning environment for the program positively. The "learning experiences" scale measures the efficacy of the general instructional approach and curriculum throughout the program. The learning environment in each part of the program was rated between "moderate help" (3.0) and "great help" (5.0). They were generally satisfied with the teaching strategies used throughout the program. For instance, 8 out of 9 found the "instructional approach taken in this class" to be "moderate" to "great" help. An equal number felt that their learning was enhanced by the way that the lectures, hands-on practices, and assignments fit together. Figure 4 documents the answer means for the learning experience. The lowest rating (mean 3.4) was given for the pace of the program.

X. Conclusions

At the beginning of this program we were uncertain about the very possibility of conveying this kind of material in an effective way to students with a limited background in mathematics, hardly any in computer science, and a still developing scientific maturity. We feared that we might be asking too much from them, and that at least some might grow frustrated and drop out of the program. We were also apprehensive about the possibility that some might find the

material arid and boring and leave for that reason. Much to our delight, these apprehensions proved unwarranted. Our students became quickly and very deeply engaged in the program and seemed to find it interesting and enjoyable. A few times they found the pace too fast and they had some suggestions on the order in which the topics were covered but, overall, they had no substantial complaint. Machine learning and molecular dynamics proved of great interest to them and they suggested that the C programming instruction could be cut down to leave room for exposure to Python.

Our conclusion is that the approach that we have taken can be successful not only in growing the scientific workforce, but also in helping undergraduates to gain a better understanding and appreciation of mathematics and its applications to science. Our students left the program highly motivated to remain in their fields and to continue their education beyond the somewhat limited horizon of a Bachelor's degree. We are particularly gratified by the fact that, on the basis on her participation in the our program, one of our students was awarded an undergraduate research assistantship in computational physics for the Fall 2017 semester.

We realize that the experience gained in a single summer with a small number of students has a somewhat limited value. However, the success of the initiative motivates us to continue it in the future gaining additional data and experience. We also recognize that a program such as ours could not be offered to large numbers of students not only because of the significant resources that this expansion would require, but also because it would probably not appeal to "average" students but only to motivated ones. Thus, for example, the same approach followed in a required course may not be as successful. On the other hand, our experience shows that the program has a major impact on the thinking and future plans of the students that can be reached and this, in our view, makes the effort very much worth while.

Acknowledgment

This project was supported by the Center for Advanced Computing and Data Science (CACDS) of the University of Houston.

References

Bound, J., Lovenheim, M. F. & Turner, S. 2010 Why have college completion rates declined? An analysis of changing student preparation and collegiate resources. *Am. Econ. J.– Appl. Econ.* **2**, 129–157.

Gepner, P., Fraser, D. L. & Kowalik, M. F. 2007 Performance evolution and power benefits of cluster system utilizing quad-core and dual-core intel xeon processors. In *International Conference on Parallel Processing and Applied Mathematics*, pp. 20–28. Springer Berlin Heidelberg.

Gepner, P. & Kowalik, M. F. 2006 Multi-core processors: New way to achieve high system performance. In *Parallel Computing in Electrical Engineering, 2006. PAR ELEC 2006. International Symposium on*, pp. 9–13. IEEE.

Guillén, P., Barrera, J., Martínez de Pisón, F., Argáez, M. & Velázquez, L. 2012 Data mining in the process of localization and classification of subcorticals structures. In *EATIS Conference Proceedings*. Valencia.

Guillén, P., Martínez de Pisón, F., Sánchez, R., Argáez, M. & Velázquez, L. 2011 Characterization of subcortical structures during deep brain stimulation utilizing support vector machines. In *33rd Annual International Conference of the IEEE*.

Malcom, S. & Feder, M. 2016 *Barriers and Opportunities for 2-Year and 4-Year STEM Degrees*. Washington DC: National Academies Press.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. 2011 Scikit-learn: Machine learning in Python. *J. Machine Learn. Res* **12**, 2825–2830.

PISA 2016 *Low Performing Students: Why They Fall Behind and How To Help Them Succeed*. Paris: OECD Publishing, http://dx.doi.org/10.1787/9789264250246-en.

Seymour, E., Wiese, D., Hunter, A. & Daffinrud, S. M. 2000 Creating a better mousetrap: On-line student assessment of their learning gains. In *National Meeting of the American Chemical Society*.

Yue, H. & Fu, X. 2017 Rethinking graduation and time to degree: A fresh perspective. *Res. High Educ.* **58**, 184–213.