Procedia Computer Science

Volume 51, 2015, Pages 1937–1946

ICCS 2015 International Conference On Computational Science

DNA and 普通話 (Mandarin): Bringing introductory programming to the Life Sciences and Digital Humanities

Mark D. LeBlanc[1] and Michael D.C. Drout[2]
[1]*Computer Science,* [2]*English*
Wheaton College, Norton, MA, USA
{*mleblanc, mdrout*}*@wheatoncollege.edu*

Abstract
The ability to write software (to script, to program, to code) is a vital skill for students and their future data-centric, multidisciplinary careers. We present a ten-year effort to teach introductory programming skills in domain-focused courses to students across divisions in our liberal arts college. By creatively working with colleagues in Biology, Statistics, and now English, we have designed, modified, and offered six iterations of two courses: "DNA" and "Computing for Poets". Larger percentages of women have consistently enrolled in these two courses *vs.* the traditional first course in the major. We share our open source course materials and present here our use of a blended learning classroom that leverages the increasing quality of online video lectures and programming practice sites in an attempt to maximize faculty-student interactions in class.

Keywords: programming, interdisciplinary, multidisciplinary, genomics, bioinformatics, digital humanities, blended learning

1 Introduction

Teaching novices to understand, predict, and solve data-rich problems by writing software is an interdisciplinary endeavor. Whereas "computational science is the new scientific field emerging from the fusion of mathematics and information technology" (Koumoutsakos, 2014), there exists a heightened need for departments to offer creative and appropriate courses that acknowledge the need for students to be exposed to interdisciplinary teams and computationally-rich problems. While programming is not computational thinking, introductory courses that teach problem solving via scripting are important course offerings that "teach programming to enhance computational thinking" (Falkner, 2014).

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2015

doi:10.1016/j.procs.2015.05.458

One goal is to offer programming courses that match the passions of rising computational scientists. We present a ten-year multidisciplinary effort to iteratively design and teach two introductory programming courses, one for students in the life sciences and the other for students in the humanities. These two introductory programming courses are both offered every other year in addition to our traditional introduction to computer science course that is offered each semester. After commenting on the spirit behind a computational thread between DNA (the language of life) and Mandarin (here just an example of a digitized corpora in most any language), we discuss the larger academic framework of these two interdisciplinary courses and present enrollment data that speaks to our efforts to increase the percentage of women who are exposed to computational science. Finally, we present details of each course, including a discussion of how our use of blended learning is helping us maximize faculty-student interactions during class and is helping to challenge the traditional notion of how faculty and students define and recognize "classroom time".

2 A note on DNA and 普通話 (Mandarin)

Counting character and word n-grams is an important step in many computational explorations of texts where vectors of token frequencies are used as "stock" in analyses of those texts; for example, an unsupervised cluster analysis of segments from a novel written by two authors. Of course, it turns out that counting "words" (motifs) in DNA (e.g., counting frequencies of motifs in a sliding window of every four nucleotides, each token referred to as a 4-mer) to detect regions of horizontal transfer is algorithmically similar to counting character n-grams in languages with little white spacing (e.g., counting instances of every four contiguous characters in *The Dream of the Red Chamber*, one of the four classical Chinese novels written in Mandarin). Although counting tokens is only one of many introductory techniques in text mining, we have found that teaching students to write software to count and store "words" for future analysis sparks their interest in computational experiments, whether that interest be in microbial genomics or text mining in a foreign language such as Mandarin.

3 Connecting Across Campus

The Wheaton College curriculum is centered on "Connections," pairs of linked courses that connect significantly different disciplines. Wheaton is a residential, liberal arts campus of 1600 students where courses are linked across any two of six academic areas: creative arts, humanities, history, math and computer science, natural sciences, and social sciences. Each course in a pair of connected courses may be taken in either order and do not need to be taken in consecutive semesters (LeBlanc *et al.* 2009). Our Computer Science program has established a suite of six connected courses, two of which are discussed here in this paper and listed in Table 1.

Course Name	Connected to Area	Connected With Courses
Computing for Poets	*English, Digital Humanities*	J.R.R. Tolkien *or* Anglo Saxon Literature
DNA	*Life Sciences, Philosophy*	Bio-Ethics *or* Ethics

Table 1. Two domain-focused, introductory programming courses for students in the humanities and life sciences. Two English courses are connected to the Poets course, and Philosophy's two ethics courses are connected to the DNA course. Ethics is also connected to other Biology courses.

Unlike models that rely on "courses outside of computer science" (Furst *et al.* 2007), connected

courses involve multidisciplinary faculty in the design of computer science courses. Humanities students, typically from the highly enrolled English program who are taking either "Anglo-Saxon Literature" or "J.R.R. Tolkien" courses are encouraged to consider completing a connection by subsequently enrolling in "Computing for Poets" (COMP 131). Given the explosion of digitized texts and our own ongoing research (*cf.* Lexomics), this course connection is rich with opportunity for creative problem solving. Likewise, the revolutions in personalized medicine and genome sequencing are generating a new thirst among students to computationally consider how "the stuff of life" is data. The "DNA" course (COMP/BIO 242) is cross-listed as a computer science or biology course and counts for a 200-level elective in bioinformatics, biology, or computer science. The "DNA" course is aptly connected with Philosophy's Bio-Ethics or Ethics courses; the fast-paced changes in genomic medicine alone keep us all on our toes regarding the implications of the scientific advances.

Overall, our experience supports the recommendation that faculty from other departments take an active role in course development and delivery (Guzdial, 2009). We agree with Cooper and Cunningham (2010) that offering opportunities for problem solving and introductory programming with a specific context over an entire semester is an important element, whether the context is "genomics" for life science students or "text mining" for digital humanities students. Introductory programming courses with a focus on media have been very successful (*cf.* Guzdial, 2003). Union College offers multiple perspectives when approaching CS1 (Barr, 2012) as do our courses that reach two vibrant and broad audiences: the life sciences and digital humanities. The team-taught course at Harvey Mudd College (Dodds *et al.*, 2012) that integrates the first semesters of BIO1 and CS1 is more ambitious than our DNA course described here but is similar in spirit in that the entire semester is focused on modules that apply computing to genomics problems. Bioinformatics-centric courses are more likely to appear at the upper-level (*cf.* Tjaden, 2007); whereas, the courses described here assume no prior programming experience. A growing number of CS1 courses offer "data-centric" assignments with topics and applications that vary on each assignment (e.g., Anderson, *et al.* 2015) whereas the DNA and Poets courses discussed here maintain a fixed context. Perhaps most distinguishing is that the two courses presented here are introductory programming courses. Although a rich collection of web-based bioinformatics and text mining tools are available and are integrated during final projects, the focus is on teaching good introductory programming skills to mix, mash, and morph scientific data within a context of an existing scholarly and scientific passion.

4 Towards gender balance

Increasing the percentage of women who enroll in introductory computer science courses has been an ongoing focus and continual challenge. In twenty offerings from 2000 to the present, our traditional first course in the computer science major (CS1 - "Robots, Games, and Problem Solving"), which itself continues to be oversubscribed each semester for the last few years, enrolls on average just over one-third female students (35.7%). Like at many institutions, this course serves potential computer science majors and as an elective for primarily mathematics and science majors. With an overall campus female-male student ratio of almost two to one and the relatively high proportion of women majoring in the life sciences and humanities, we knew we had positive recruitment potential if we could provide more options.

Since 2004 we have offered two additional introductory programming courses: "DNA" for students in the life sciences and a course called "Computing for Poets" (hereafter Poets) for students in the humanities. The two courses are separately offered as an alternative to our introductory (CS1) course, but specifically target the rapid need for and reliance on computational

thinking due to the revolutionary changes in the bioinformatics and digital humanities spaces. Over the last decade, DNA has been offered six times and Poets has been offered five times since 2004.

As shown in Figure 1, our two introductory, interdisciplinary courses (DNA and Poets) have consistently enrolled more women than the traditional offerings of CS1. In all six offerings of DNA and in all five offerings of Poets since 2004, women enroll in higher percentages than 19 of the 20 offerings of CS1. Since 2000, women enroll in CS1 at an average of 36%. In contrast, from 2004 to the present, on average 51% of the students in DNA and 58% of the students in Poets were women.

Maintaining female enrollment percentages above 50% in introductory computer science courses is a significant and refreshing outcome. Greater gender balance in the classroom has numerous benefits, including the benefits that the male students receive from working more closely with women early in their careers.

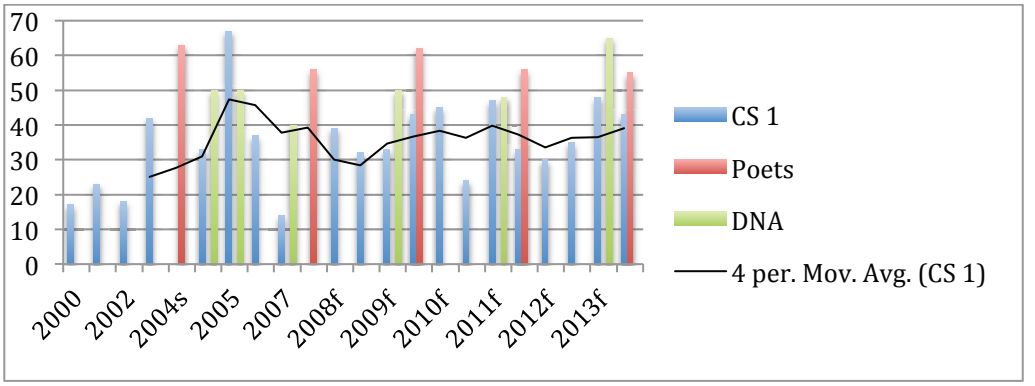

Figure 1. Percentage of Women Enrolling in three different introductory computer science courses: CS1 (introductory course for majors) and two focused offerings, one for students in the life sciences (DNA) and the other for students in the digital humanities (Poets). The moving trend line shows the 4-per-year average percentage for our traditional course (CS1).

5 The Digital Humanities

The time has never been better for computational science to impact the Humanities. The Digital Humanities represents a growing subculture on many campuses and the glut of digitized texts, many in languages from around the globe that match scholar's areas of expertise, has radically altered what it means to be a scholar of texts. Computational explorations of texts, sometimes referred to as computational stylistics is a subfield within the Digital Humanities and one where students need exposure to and practice with searching and analyzing large digitized corpora. The Computing for Poets course discussed next is approaching its sixth iteration to do just that.

5.1 Computing for Poets (COMP 131)

The use of computers to manage the storage and retrieval of written texts creates new opportunities for scholars of ancient and other written works. Recent advances in computer

software, hypertext, and database methodologies have made it possible to ask novel questions about a poem, a story, a trilogy, or an entire corpus. The Poets course exposes students to leading markup languages (HTML, CSS, XML) and teaches computer programming as a vehicle to explore and "data mine" digitized texts. Programming facilitates top-down thinking and practice with computational thinking skills such as problem decomposition, algorithmic thinking, and experimental design, topics that humanities students in our experience rarely see. Programming on and with texts introduces students to rich new areas of scholarship including stylometry and authorship attribution. The course has no prerequisites other than a love of the written (and digital) word; no previous computer programming experience is required.

A learning objective for students in this course is to articulate how computational analyses of digitized texts enables both a "close reading" of a single text and as well as a "distant reading" of many texts across time (Moretti, 2013). The goal for each student is to master enough programming to modify digitized texts to help in a computational experiment that explores a question of a text or set of texts. For example, the assignments in Fall 2013 (see Table 2) asked students to write and extend Python scripts over the semester that analyze texts and store results in Excel-ready output files to facilitate subsequent analyses, documentation, and scientific writing. The number, pace, and level of difficulty of assignments and labs in the course are coordinated to help most students conduct an introductory text mining experiment in the last three weeks of the course.

Assignment	Short Description
a1: Website – Ngrams	Build a website with results from Google's Ngram Viewer to investigate how the frequency of words or phrases have changed over time as appearing in books from 1800 to 2012
a2: Deforming Poetry	Write a program to help the reader more easily read poems that have been deformed in various methods, e.g., read a poem backwards
a3: Regex Play	Use regular expressions to solve some of Will Shortz' word puzzles; Shortz is National Public Radio's (NPR) puzzle master
a4: Tall Elves	Conjecture: Tolkien wanted his readers to fully appreciate that his elves were large, thus he used the word "tall" (or other variants such as "big", "giant", "large", etc.) in close proximity to the name of an elf (e.g., "Legolas", "Galadriel" or even the generic word, "elf"). Write a script(s) to generate data that will help experimentally verify if this conjecture is true or false.
a5: Only in the Poetry	Considering the entire digitized corpus of Old English (Anglo-Saxon) poetry and prose texts, write a script to determine which if any words appear only in the poetry?

Table 2. Five programming assignments in Computing for Poets (Spring 2014 semester)

Developing programming assignments *with* an expert scholar in these spaces is critical when attempting to focus student attention on the current level of scripting practice and excitement that comes from studying the original texts. The Poets course is "connected" with two courses in English: J.R.R. Tolkien (ENG 259) and Anglo-Saxon Literature (ENG 208). For example, in Anglo-Saxon studies, the relationship between Old English poems has been a vexed question for nearly 150 years. Most of the poetry is anonymous and exists only as tenth-century copies in manuscripts (some of it is assumed to be much earlier). We have only three named authors of poetry in the Anglo-Saxon period and the majority of the prose is anonymous. Thus for years scholars of Old English have struggled to divine relationships between texts based on vocabulary,

meter, and style. These results have been at best contentious and at worst completely unsuccessful. As students learn more and more scripting in the course, we set up and run experiments using the entire Anglo-Saxon corpus. Some of the questions asked by undergraduates may never have been asked before. In the case of the connection with the course on J.R.R. Tolkien, students participate in the design and execution of an experiment across multiple texts, e.g., The Lord of the Rings trilogy which inevitably leads to follow up interests with other texts: What about the *Silmarillion*? Should we ask this computational question on *The Hobbit*?

5.2 Blended Learning

Acknowledging that class time is precious, we have worked hard to maximize student problem solving practice in class. In a modest use of blended learning in a format sometimes called a "flipped classroom" approach, students use an online Python interactive textbook (Miller and Ranum, 2014) and spend time outside of class completing the online Code Academy practice exercises. The level of quality in both the interactive textbook and practice lessons at Code Academy is notable. Together these provide helpful benchmarks for student progress, helping ensure that students have practiced with the fundamental control structures, for instance, before participating in a hands-on lab. It has not escaped our notice that our open reliance on online reading and practice materials has forced us as instructors to critically consider the use of class time, for example, the time spent "writing notes on the board." In-class problem solving sessions begin with brief discussions that map a problem at hand to the programming language control structures and/or data structures of the day, followed by pair-programming opportunities (switching the student typing every 20 minutes) to refactor scripts to open and read from files in multiple languages, e.g., Mandarin, Latin, or Middle English.

5.3 Final Projects

From the initial day of class, students begin work on a final semester project to design an experiment on a set of digitized texts of their choice. In our experience, scholars who might like to perform computational analysis in their areas of expertise and/or wish to teach their students how to do so become discouraged too early in the game. The *Lexos* software developed by our Lexomics Project provides a simple, web-based workflow for text processing, statistical analysis, and visualization. Situated within a clean and simple interface, *Lexos* consolidates the common yet frustrating pre-processing operations that are needed for subsequent analysis, e.g., a cluster analysis of segments from multiple novels. Student programming in final projects is on an "as needed basis," the focus being the use of a computational method in a small experiment. Some recent undergraduate topics for final projects are listed in Table 3. A complete syllabus and sets of programming assignments and other course materials are available at our Lexomics website (http://lexomics.wheatoncollege.edu/).

Sample Undergraduate Final Projects in Computing for Poets
All of Caesar is Divided into Five Parts, but Who Wrote What? A look at the Various Authors of the Complete Works of Julius Caesar
Not So Elementary: Did Sir Arthur Conan Doyle Write all of the Sherlock Holmes Canon?
Found in Translation: A Comparative Lexomic Analysis of Three Translations of Beowulf
Variation and Influence Using Dendrograms to Identify Variations in Style and Influence in Tolkien's *Lord of the Rings*
Can Bias be Counted? Political Vocabulary in the News

Table 3. Student topics for their final projects in Computing for Poets (the last three iterations of student topics are available at: http://wheatoncollege.edu/lexomics/computing-poets).

6 Reaching the Life Sciences

The generation, storage, analysis, and visualization of bioinformatics data is happening at a dizzying pace and the next generation of scientists face great challenges in a post-sequenced world (Macarthur, *Wired*). Since 1998, our Wheaton College Genomics Research Group has influenced collaborative teaching between biology and computer science and our collaborations have in turn shaped directions for research (LeBlanc and Dyer, 2003; 2004; Dyer *et al.* 2007).

6.1 DNA (COMP/BIO 242)

An amazing blend of science, computing, and mathematics emerges when considering the molecule "Deoxyribonucleic Acid" (DNA). DNA is the blueprint of life for all organisms on Earth. Its distinctive and beautiful physical nature, a double helix of four bases, maps onto its functionality as a bearer of information, generation after generation. Fully sequenced genomes including the human genome and hundreds of microbial genomes have become the starting point for attempts to answer a wide range of biological and quantitative questions. A goal of the course is to enhance computational thinking via introductory programming as applied to the wealth of genomic data. A particular focus is on the exciting merge of personalized medicine and the ongoing human microbiome projects. Table 4 lists a set of learning objectives, what we hope become our students' "take away stories".

"Take away stories" (learning outcomes) for DNA (COMP/BIO 242)
(0) You are at a cocktail party and the topic of genomes comes up. You are able to recall significant phrases, terms, and techniques and your understanding of the main ideas and concepts enables you to lead the conversation for a while … which causes your friends to raise their eyebrows.
(1) You learn to identify and classify problems that are candidates for a computer to handle; this is the start of "computational thinking".
(2) You demonstrate the ability to think algorithmically, breaking what originally seems like an overly complicated problem into a series of smaller, manageable tasks.
(3) You learn to craft creative solutions by "writing software" ("to program", "to code", "to script").
(4) You appreciate the importance of microbes and the Human Microbiome Project.
(5) You design experiments to first solve small computational tasks (e.g., one gene sequence) and then scale your solutions to very large sets of data (e.g., all genes in a genome).
(6) You learn to move around and perform some work in the Linux (Unix) operating system.
(7) You learn to professionally document your software and produce quality summaries, graphs, and reports of your computational methods and results.
(8) You begin to appreciate the (soon to be) revolution in personalized medicine, including knowing your way around a personalized report from the personal genomics company "23andMe".
(9) You feel empowered to evaluate the ethical implications of your work and learn to appraise, critique, and defend your own as well as the work of others.

Table 4. Learning objectives for the cross-listed course DNA (COMP/BIO 242).

This course is part of the connection "Genes in Context" with Philosophy 111 (Ethics) or Philosophy 241 (Bio-Ethics). Throughout the semester in the DNA course, students are exposed to the ethical aspects of living in a post-genomic world and the increasing use and challenges of sequenced genomes as applied to "personalized medicine". Students access, explore, and discuss

the professor's genomic report as obtained from the saliva-based DNA service (In Fall 2013 we obtained one of the last reports to include likelihoods of having certain diseases, just prior to the FDA's decision to halt such "interpreted" genomic profiles). In addition, students watch the 1997 movie GATTACA together, a bioethics professor leads a discussion of "Designer Babies", and students produce one-minute YouTube "commercials" of companies currently promoting and selling medical profiles based on individual genomes. The commercials are framed from one of two points of view: (i) from the point of view of the company (e.g., 23andMe) or (ii) from a consumer advocacy point of view.

Not unlike the Poets course, the programming assignments are paced so that students may conduct a final project experiment and then share their methods and results, both orally and in writing.

Assignment	Short Description
a1: Playing with DNA	Working with DNA as a language: a string of characters in a four-letter alphabet
a2: Chargaff's Numbers	For any sequence or entire genome, report the proportions of `A,C,G,T` nucleotides
a3: Gene Finder	Simulate transcription and translation on strings and evaluate "appropriate" reading frames
a4: Motif Finder	Build and apply regular expressions to find relevant regulatory motifs upstream of genes
a5: Comparative Genomics	Use a "bag of words" to keep track of motif frequencies to assign a "genomic signature" to sections of a genome

Table 5. Five programming assignments in DNA (Fall 2013 semester).

6.2 Blended Learning

The Fall 2013 offering of the DNA course was our most significant blended learning trial to date. Like in the Poets course, students use an online Python interactive textbook (Miller and Ranum, 2014) and spend time outside of class completing the Code Academy practice exercises. In addition, students watch at least five lectures outside of class on biological topics from a MOOC (Massive Open Online Course), here Udacity's "Tales of the Genome" online course. The rationale for using this "outside" material is two-fold. First, the quality of these terminology-rich lectures is very good and improving; for example, students are encouraged to fill-in a template of a concept map during a lecture, where each concept map links to and from other lectures. The online material contains minimal "talking head" time, rather presenting a series of "whiteboard" illustrations, punctuated by just-in-time quizzes. A completed concept map is provided at the end of the lecture. The second and related rationale for using Udacity's materials is that class time is too precious to spend lecturing on basic biological processes, for example transcription and translation, especially in a one-semester programming course where time for problem solving is at a premium.

7 Reaching the entire academy

Teaching programming to introduce and enhance computational thinking is a vital contribution to the academy. The life sciences' genomic revolution and the digitization of books and manuscripts present unique opportunities for exposing a wider audience to computational science. We present two introductory programming courses over a ten-year period that target students in the life sciences and humanities, each attracting gender balanced enrollments that exceed traditional computer science introductory offerings. In particular, we remain convinced that learning to think algorithmically and to encode those ideas in software is a vital competency for today's undergraduate. New opportunities for teaching computing to the growing constituencies that can benefit from introductory programming have led us to experiment with alternative, blended learning uses of class time, specifically fewer minutes lecturing and more hours solving problems in class. We advocate for more creative experimentation from faculty with how they use class time, including an increase in the infusion of "outside" course materials (e.g., MOOC lectures) from the growing palette of good instructional materials available.

Programming is not an end all for computational science. Yet, the academy faces a number of new audiences who will benefit from the ability to script in the midst of their data-driven world. How we reach them in introductory courses can make all the difference.

This work was funded in part by the National Endowment for the Humanities (NEH) and a Google CS Engagement Award.

References

Anderson, R., Ernst, M.D., Ordóñez, R., Pham, P., and Tribelhorn, B. (2015). A Data Programming CS1 Course. *Proceedings of SIGCSE'15 Symposium on Computer Science Education*. Kansas City, MO (Mar. 2015), 150-155.

Barr, V. (2012). Create two, three, many courses: An Experiment in Contextualized Introductory Computer Science. *JCSC* 27(6) (June 2012), 19-25.

Code Academy: Python course (accessed 12/1/2014). (http://www.codecademy.com/en/tracks/python).

Cooper, S. and Cunningham, Z. (2013). Success in Introductory Programming: What Works? *Communications of the ACM.* August, 56(8), 34-36.

Dodds, Z., Libeskind-Hadas, R., and Bush, E. (2012). BIO1 as CS1: Evaluating a Crossdisciplinary CS Context. *ITiCSE'12*, July 3–5, 2012, Haifa, Israel, 268-272.

Downey, S., Drout, M., Kahn, M., and LeBlanc, M.D. (2012). "Books Tell Us": Lexomics and Traditional Evidence for the Sources of *Guthlac A*. *Modern Philology* 110, 153-181.

Dyer, B.D., Kahn, M., and LeBlanc, M.D. (2007). Classification and regression tree (CART) analyses of genomic signatures reveal sets of tetramers that discriminate temperature optima of archaea and bacteria. *Archaea* 2, 159–167.

Falkner, N. (2014). Computational Thinking for All. Keynote address at *SIGCSE Technical Symposium on Computer Science Education*, Atlanta, GA.

Furst, M., Isbell, C., and Guzdial, M. (2007). Threads™: How to Restructure a Computer Science Curriculum for a Flat World. *Proceedings of 38th SIGCSE symposium on Computer Science Education*. Covington, KY (Mar. 2007), 420-424.

Guzdial, M. (2003). A Media Computation Course for Non-Majors, ITiCSE 2003 Conference Proceedings, 104-108.

Guzdial, M. (2009). Teaching Computing to Everyone. *Communications of the ACM*, 52, 5 (May 2009), p31-33.

Koumoutsakos, P. (2014). The Arrow of Computational Science. Abstract of public lecture given at ETH Zurich, Switzerland, June 2, 2014.

LeBlanc, M.D. and Dyer, B.D. (2003). Teaching Together: A three-year case study in genomics. *The Journal of Computing in Colleges*, 18(5), 85-95.

LeBlanc, M.D. and Dyer, B.D. (2004). Bioinformatics and Computing Curricula 2001 – Why Computer Science is well positioned in a post-genomic world. *ACM SIGCSE Bulletin*, 36(4).

LeBlanc, M.D., Gousie, M., and Armstrong, T. (2010). Connecting Across Campus. *Proceedings of the 41st SIGCSE Technical Symposium on Computer Science Education*, Milwaukee, WI.

Lexomics Research Group. http://wheatoncollege.edu/lexomics/computing-poets/

Macarthur, D. (accessed 12/09/2014). Why biology students should learn to program. *Wired: Genetic Futures. http://www.wired.com/2009/03/why-biology-students-should-learn-how-to-program/*

Miller, B. and Ranum, D. (accessed 12.10.2014). How to Think Like a Computer Scientist. Runestone Interactive Project (http://interactivepython.org/courselib/static/thinkcspy/index.html).

Moretti, F. (2013). *Distant Reading*. Verso Publishing.

Tales of the Genome: An Introduction to Genetics for Beginners. Udacity. (https://www.udacity.com/course/bio110).

Tjaden, B. (2007). A multidisciplinary course in computational biology. *The Journal of Computing in Colleges* 22(6), 80-87.

Related URLs

Computing for Poets:	http://wheatoncollege.edu/lexomics/computing-poets/
DNA:	http://wheatoncollege.edu/genomics/dna/
Lexomics Research:	http://lexomics.wheatoncollege.edu
Genomics Research:	http://genomics.wheatoncollege.edu