Annotated Bibliography Assignment

Jonathan St-Onge

Table of contents

Introduction

As science becomes increasingly collaborative and data-driven, scientists now need to know how to store, manage and move their data in a reproducible and transparent way. In many fields, communities of scholars propose new research programs based on computational methods to improve data-driven inquiries. In this project, we are interested in the perceived costs and benefits of students when learning to code. For instance, software development is typically perceived as a male (and geeky) activity. Does this mean that it is more difficult for women to assume the role of programmer? Does having peers participating in programming projects decrease the perceived cost of learning to program? We review the literature on the (i) rise of programming in non-Computer Science (non-CS) majors, including (ii) the underlying reasons why people advocate for programming in those fields, and (iii) what lessons have already been learned when teaching programming in non-CS majors.

The rise of programming in non-CS fields

Computational X

> description
>
> In the last 20 years, we witnessed the datafication and digitalization of many fields that are not traditionally informed by computer science. History gave us four main disciplines representing this programming shift; digital humanities, cultural analytics (media computation), computational social science, and arguably computational linguistics (not including information systems).

Hockey, S. (2004), *The History of Humanities Computing*

> Seminal article describing how different groups of researchers came to use computing to answer questions in the humanities. Digital humanities include fields such as history, literature, journalism, philosophy, and other archival studies. Topic modeling is a key computational method used by digital humanists.

Wing, J. M. (2006), "Computational Thinking"

Lazer, D. et al. (2009), "Computational Social Science"

> Computational social science seems to be closer to the natural science in how they handle, in particular physics. Computational social sciences includes fields such as sociology, political theory, economy, and traditional sciences who got into studying social media. Computational social science tends to be closer to big data analysis, with observations being more easily reduced to quantitative data than in digital humanities or cultural analytics.

Manovich, L. (2009), "Cultural Analytics"

Alvarado, R. C. (2012), *The Digital Humanities Situation*

Lazer, D. M. J. et al. (2020), *Computational Social Science*

Edelmann, A. et al. (2020), "Computational Social Science and Sociology"

Review of computational method in sociology. The authors are practitioners interested in the rise of computational methods in social science with a focus on sociology. They use the Web of Science database and citation network analysis to show the significant increase in computational methods in social science. They argue that computational sociology enables the study of previously unavailable questions, provides new ways of creating theory itself, and facilitates the theorizing of digital behaviors. Although they claim that the calculation methods are on the rise, they do not provide solid evidence for this. They make their claims simply based on descriptive statistics from observational data.

Positive vibes only: learning to code is nice and even necessary

Forte, A. (2003), *IEEE Symposium on Human Centric Computing Languages and Environments, 2003. 2003*

Similar to Kennedy (2017), Forte argues that the inability of computer science programs to bring on board diverse groups could lead to computational illiteracy in those groups. As such, the benefits of learning to code in diverse groups go beyond technical skills to encompass an ethical duty. Forte (2003) finds that teaching programming as part of another discipline (in her case media computation) can help motivate students from groups excluded from computer science.

Kennedy, K. (2017), "A Long-Belated Welcome"

Given the growing number of women in the humanities, Kennedy (2017) argues that fostering digital skills is a feminist imperative and ethical duty. To incorporate computational methods in classrooms, she proposes that teachers should focus on developing hybrid digital-humanities methods, such as exercising students' ability to judge the relevance of online content, exploiting multimodal and interactive documents, or practicing moving between the macro and micro reading perspectives. Kennedy offers an original perspective on how the teaching of digital literacy skills should be motivated from a variety of perspectives, including that of feminist imperatives and diversity ideals. But the article contains many normative statements about the practices we should adopt without necessarily taking into account all the practical challenges involved.

D'Ignazio, C., & Klein, L. F. (2020), *Data Feminism*

Edelstein, D. et al. (2017), "Historical Research in a Digital Age"

Similar to Edelmann et al. (2020), Edelstein et al. (2017) provide a review of how digital methods open up new methodological avenues for historical inquiries. This article is less of a manifesto for digital humanities than showing how to handle the exponential growth of new information (big data) in the authors' 10 years project of mapping the Republic of Letters. Edelstein et al. (2017) put forth the challenge of designing a workflow that "support humanistic inquiry into multidimensional, heterogeneous, and incomplete datasets" (419). They conclude that new digital sources in history call for new tools that can accommodate ambiguities, paradoxes, and contingencies that are specific to digital history. Ultimately, the paper is high-level and does not address the challenge of interdisciplinary collaboration (historians, grant officers, data scientists, programmers, and interface designers). The benefits they identify are based on their own experience and not their object of study.

Heuser, R., & Le-Khac, L. (2011), "Learning to Read Data"

Heuser and Le-Kac (2011) show how a digital analysis of 19th-century British novels informed by humanistic values can address the methodological concerns of traditionalists in the humanities. Based on an inverse relationship between two semantic fields, they show how a quantitive signal can be enriched by concepts from the humanities (such as the 'death of values in the Victorian period') to recover units familiar to literary scholars (novels, genres, and authors in a corpus). The authors exemplify how digital and traditional tools mutually inform each other to achieve an analysis of a large corpus that remains true to the values of the humanities. The authors do not address the potential asymmetry that traditional scholars face when digital humanists make many of the decisions that typically arise while cleaning the raw data, which influences the output that traditional scholars must analyze.

Teaching programming to non-CS major

description

Numerous articles report on teaching programming to non-Computer Science majors. Early papers are written by folks in computer science who observed that non-CS majors disliked and tended to fail traditional introductory programming courses such as CS1. As departments and teachers started to offer classes on particular topics that required programming skills, we started to see papers on individual experiences of teaching programming to students who might or might not care about programming. Overall we can say that teaching programming for programming's sake does not work outside computer science. Non-CS majors need to first see the potential benefits of learning to code before enjoying it. A recurrent theme is how novices require mentorships and peers to succeed in carrying out projects that require programming. The precise arrangement of which expertise level each actor needs when doing a project to maximize the effect of learning programming is yet to be determined.

Chilana, P. K. et al. (2016), *CHI'16*

Study about motivations of CS vs non-CS majors when taking introductory class in programming.

Dawson, J. Q. et al. (2018), *SIGCSE '18*

This study performs a quasi-experiment design in which non-CS major students took the primary introductory computer science (CS1) course one year while the next year they took the same class but designed for other disciplines (CS0.5). The key difference between the two conditions is that the workload in CS0.5 was geared towards solving problems from a discipline of their choice, while students in CS1 pursued more advanced topics in computer science. Follow-up surveys demonstrate that students in CS0.5 performed better and were more satisfied than their peers from the previous year who took CS1. As in Kafura et al. (2015), the survey reveals that a recurrent theme that explained higher satisfaction for CS0.5 students was the ability to work on relevant problems using code with peers of their choice. One downside of the experimental design is that there might have been biased introduced by using two cohorts from two different years.

Kafura, D. et al. (2015), *Design and Preliminary Results from a Computational Thinking Course*

Kafura et al. (2015) shared their experience of building a new introductory course in computational thinking for non-computer science majors. After giving the class in Fall 2014, they perform a survey with the students (n=20; 70% male and 30% female) about their pedagogy. One tool they put to test is Blockly, a block-based programming language that let students organize code as modular blocks, which can be translated into a popular textual programming language like Python thereafter. Follow-up interviews revealed that working in a cohort was beneficial for students, providing peers to go see when stuck on a programming problem. That said, their results should be taken with a grain of salt as their sample is small and students chose to enroll in a computational thinking course (purposive sampling).

Marsh, A. C. (2013), "Omeka in the Classroom"

Marsh (2013) reports a snapshot of her 10-year project that seeks to instill programming interests in graduate students in museology. Seeking to confront her student with the challenges of representing physical objects in a virtual environment, a new reality of museum curators, Marsh tasks her students to create a digital exhibit using the open source software Omeka. With this assignment, she finds that most students recognized the subtlety of working in the digital format, but also that the limitations of the Omeka platform have led more curious students to engage in improving Omeka. While Marsh's project may be unrepresentative of other disciplines, her experience contributes to the growing number of educators exploring the challenges of teaching digital skills to digital natives, who may have little interest in the digital world as part of their professional training.

O'Sullivan, J. et al. (2015), "Programming in the Digital Humanities"

O'Sullivan et al. (2015) took advantage of the Digital Humanities Summer Institute to survey participants (n=96) on their use of programming and the role of software development in the humanities. Their results suggest that there is a slight generational gap between senior researchers (50+) and their younger counterparts (25-35), with the younger generation not considering themselves "technically proficient" and preferring to collaborate rather than coding everything on their own (unlike older researchers). As the authors acknowledge, we must be careful in interpreting their results because the survey is too small and potentially unrepresentative of the larger population. Nonetheless, as with Marsh's project (2013), their findings contribute to the growing sense in the digital humanities that humanists and technical experts need to find ways, or middle ground, to collaborate effectively to complete computational projects.

Medeiros, R. P. et al. (2019), "A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education"

Auker, L. A., & Barthelmess, E. L. (2020), "Teaching R in the Undergraduate Ecology Classroom"

Stroulia, E. et al. (2011), *Proceeding of the 2011 Community Building Workshop*

Stroulia et al. (2011) aim to investigate how to make undergraduate students effective in their distributed software engineering collaboration, a familiar type of collaboration for scientists. They designed the Undergraduate Capstone Open-Source Project (UCOSP), a project where students learn to code through learning and projects but in a distributed fashion. Their courses brought together 184 students from 25 universities and ran from 2008 to 2011. They find that bringing students face-to-face for a code sprint at the right time and with the right skill level greatly facilitates the distributed work over the long term. They also find that effective time management is one of the main issues in this type of collaboration. We note that the study is not experimental in that they have no control groups and no statistical tests. The authors offer their reflections on what has worked for distributed collaboration in projects requiring software engineering. This article is relevant to our study in that learning to collaborate effectively can be a benefit of learning to code in all sciences.

Jacobs, C. T. et al. (2016), "Experiences With Efficient Methodologies for Teaching Computer Programming to Geoscientists"

Teaching programming to UCL geoscience students using a blended learning approach, based on the Software carpentry methods. The study focuses on how different cohorts react to different teaching styles. They do note that motivating programming with geoscience projects was key, as was the game-style approach to do so (MOOC-style did not). Jacobs et al. (2016) provide guidelines on how to teach effectively programming to non-CS majors. They do not speak of demographics or other factors that might come into play, but they mention tool attributes to ease the learning (Ipython > Python > C).

Alnervik, T. et al. (2020), *Success Factors in an Introductory Programming Course in a Non-CS Major*

Thesis on the success factors of teaching programming to non-CS majors. Motivation (including "encouragement from others" and "keenness and general academic motivation") seems to be more important than previous math or programming experience.

Sharma, R., & Shen, H. (2018), "Does Education Culture Influence Factors in Learning Programming"

Coding is collaboration: the impact of social networks on learning

Stadtfeld, C. et al. (2019), "Integration in Emerging Social Networks Explains Academic Failure and Success"

Stadtfeld and colleagues (2019) produce a survey driven by ideas from network theory to demonstrate the impact of social integration of new engineering students on GPA. Although this is limited to a single context and success metric, this article provides some evidence that friendship networks, not just study buddy relationships, are important in quantifying failure and academic success. This paper goes beyond other papers discussing the relative importance of community in that it investigates how types of networks are key to understanding academic success. That said, the paper has limitations in that GPA may or may not be causally related to the social drivers of research productivity and relevance.

Galesic, M. et al. (2021), "Human Social Sensing Is an Untapped Resource for Computational Social Science"

Galesic et al. (2021) show that asking individuals about their expectations for voting behavior or the vaccination status of their social contacts could provide more accurate estimates of these behaviors than asking about them. They argue that people can act as social sensors in that they are acutely aware of some behaviors of their peers, contrary to the view that we are merely biased individuals. Social sensing is relevant to this study in that we aim to determine the social drivers that facilitate learning to program, which might be well predicted by the opinions individuals have of their social contacts. The authors frame their study as part of the field of computational social science, but one might wonder how different it really is from one of the more traditional social sciences such as sociology.

Werner, L. L. et al. (2004), "Pair-Programming Helps Female Computer Science Students"

It's a trap.

Social barriers to learning to code

Jackson, L. A. et al. (2008), "Race, Gender, and Information Technology Use"

Using a sample of 602 children from southern Michigan's Lower Peninsula, Jackson et al. (2008) investigate the impact of gender and race on using information technologies (IT; e.g. computer, Internet, video games and mobile phone use) and their impact on school performance. Unsurprisingly, they find evidence of a digital divide, in which Caucasian American children use/have more computer access than African American children, boys play more video games than girls, and girls use computers more for communication than boys. Knowing which population is exposed to IT early, combined with other related socio-economic biases, is key to understanding the unequal costs of using digital tools later on, especially in relation to self-identifying as programmers. A limitation of this study is the small sample size and lack of more appropriate statistical modeling (such as a mixed model), which would have allowed the authors to control for the many confounding variables that may underlie the digital divide.

Balali, S. et al. (2018), "Newcomers' Barriers. . . Is That All? An Analysis of Mentors' and Newcomers' Barriers in OSS Projects"

Balali et al. (2018) identify barriers for mentors in open source programming projects based on semi-supervised interviews, with a particular focus on gender-specific challenges. Among the 35 barriers identified, they find evidence of 11 personal barriers (e.g. switching context between mentees), 12 interpersonal barriers (e.g. creating an inclusive community, harsh project atmosphere), 8 process barriers, and 6 technical barriers (e.g. identifying the right level of complexity for newcomers). Overall, this paper appears readily applicable to academia, as new students in traditionally non-digital fields will likely encounter the same barriers to entertaining successful mentor-mentee relationships. While some aspects of this work might be applicable to academia, other dimensions such as the decentralized collaboration scheme typical in open source projects might not.

Trinkenreich, B. et al. n.d., *Women's Participation in Open Source Software*

Trinkerenreich et al. (2022) provide an exhaustive literature review from the last 20 years on why women who identified as such tend to leak out or avoid participating to open source software (OSS) communities. One key takeaway is that men tend to participate in OSS because they find it intrinsically rewarding (e.g. the joy of the craft, but also a community who think alike), whereas women are motivated to join OSS project to accomplish their goals. Trinkenreich et al. (2022) also review a number of strategies to mitigate the male-dominated culture of programming, most notably by promoting awareness of women lead, de-stereotyping language about programming, and fostering rich social environments with gender parity. The authors also make explicit the connection between the situation of women in OSS women with their participation in STEM, including the lack of parental leaves policies and difficulties to achieve work-life balance.

Forrester, C. et al. (2022), "Undergraduate R Programming Anxiety in Ecology"

Forrester et al. (2022) generalize the anxiety studied in computer science and mathematics and apply it to learning to code in the R programming language. Using a survey completed repeatedly over the course of a semester, they provide evidence that undergraduate women in ecology feel uneasiness and apprehension about learning the R programming language. This paper offers an interesting case study of the challenges to teaching programming, informed by the literature on anxiety in academia, but ultimately they rely on a small sample size and fail to carry on more sophisticated statistical analysis as a result. Still, this is a good starting point for understanding the relative costs and benefits of teaching programming to a cohort with no prior coding experience.

Connolly, C. et al. (2009), "Programming Anxiety Amongst Computing Students—A Key in the Retention Debate?"

Anderson, K. et al. (2016), *Student Labour and Training in Digital Humanities*

notes

- Most faculty researchers consider their work on these projects to be highly collaborative (FS 5), while most students consider their work to be only minimally or moderately collaborative (SS 18).
- While nearly half of faculty members' projects had an anticipated duration of six years or more (FS 6), a similar number of student researchers anticipated working on their projects for two years or less (SS 7).

Posner, M. (2012), *Some Things to Think about Before You Exhort Everyone to Code*

A blog post about what learning to code entails for women. The cost of learning to code while knowing that programming has been constructed as a male

Nolan, K., & Bergin, S. (2016), *Koli Calling 2016*

Widner, M. (2012), *Learn to Code; Learn Code Culture*

skud (2011), *On Being Harassed*

Mind the gap between scientific practices and curricula

Touchon, J. C., & McCoy, M. W. (2016), "The Mismatch Between Current Statistical Practice and Doctoral Training in Ecology"

notes

- There are more and more published papers with R but learning R is absent from of curricula in ecology.

Is programming in the humanities facilitate neoliberalism?

Allington, D. et al. (2016), *Neoliberal Tools (and Archives)*

> This essay epitomizes the risks of focusing too much on programming to the detriment of domain expertise. Alling et al. (2016) argue that learning to code in the humanities is first and foremost a neoliberal scam, that is, a way to make universities profitable for corporations by drawing from external funding sources and provide students with marketable skills.

Hui Kyong Chun, W. et al. (2016), *The Dark Side of the Digital Humanities*

Locke, B. T. n.d., *Digital Humanities Pedagogy as Essential Liberal Education*

Greenspan, B. (2019), *The Scandal of Digital Humanities*

Epistemic inequality: Is science really meritocratic?

> Coming from the sociology of science and the science of science, this emerging field is interested in combining qualitative data with bibliometric studies to make claims about the unequal spread of ideas in science. We can trace back their ideas and motivation from the literature on the "cumulative advantage" in science, the "gender gap", or "invisible colleges". The field distinguished itself by its methods, which combine qualitative (surveys) and quantitative (bibliometric) data.

Piper, A. (2016), "There Will Be Numbers"

Way, S. F. et al. (2019), "Productivity, Prominence, and the Effects of Academic Environment"

Way and colleagues previously investigated how institutional prestige, or the institution's ability to place their doctoral students in faculty positions, was instrumental in understanding the unequal spread of scientific ideas. In this follow-up study, Way et al. (2019) demonstrate that a working environment that promotes larger research groups and available labor is more important than prestige to measure scholarly outputs. Using an experiment where researchers are matched based on gender, subfields, and other features, but they in with respect to their working environment, they can show that the more prestigious work environments lead to more citations even if they come from similar training environments (the contrary is not true). A limitation of this study is that their measures of scholarly outputs are limited to the number of publications and citations, which might or might not best represent the productivity and prominence of researchers.

Morgan, A. C. et al. (2021), "The Unequal Impact of Parenthood in Academia"

Morgan et al. (2021) show that parenthood in academia is a key non-meritocratic factor that influences the varying productivity of researchers, especially with respect to gender (the so-called productivity gap). Using a similar matching experiment to Way et al. (2018), but this time with parenthood being the decisive factor, they show that mothers in computer science produced on average 13.1 fewer papers compared to fathers over the same early-career time period. Their methods demonstrate how a simple survey with basic questions (e.g. who is using parental leaves and whether parent leave policies were important in choosing a current position) can provide important insights to better understand social factors driving the evolution of science. As with many natural experiments in computational social science, this study is observational and is vulnerable to confounding variables and other biases from the sampling procedure.

Wapman, K. H. et al. (2022), "Quantifying Hierarchy and Dynamics in US Faculty Hiring and Retention"

Amateur software development: No, you're not done when your code just works

Programming is on the rise in non-CS fields. To preserve diversity and minimize epistemic inequality, we want marginalized groups (e.g. women+) in CS to be on board. If not, the disciplines, research groups, and individuals who embrace programming are likely to appropriate themselves more projects requiring the use of the practices of programming. If these projects are more easily funded, this will increase epistemic inequality. But here is the annoying bit. We don't want people to code well, new projects involving programming should follow best practices. At the moment, there are reasons why we want that at the community level, but it is unclear how individuals benefit from investing time in amateur software development.

Reproducibility, reproducibility, reproducibility

Trisovic, A. et al. (2022), "A Large-Scale Study on Research Code Quality and Execution"

Trisović et al. (2022) compare shared research code in the R programming language (n=9078) and datasets (n=40,000) on the Harvard Dataverse repository with established best practices. They were interested in coding errors that lead to the impossibility of easily re-executing the code. They find that 74% of R files fail to re-execute out of the box, while 56% still fail after authors apply basic code cleaning. One of the main takeaways is that failing to produce quality code, from a software engineering perspective, leads to less reproducibility and wasted time. On the flip side, following best programming practices saves time and ensures research reproducibility for oneself and others. This study assumes that quality code should save researchers' time, but does not quantify this claim. That is, is the time invested in learning good coding practices worth the investment from a professional perspective, especially when scientists must already learn numerous other skills?

Culina, A. et al. (2020), "Low Availability of Code in Ecology"

Stodden, V. et al. (2018), "An Empirical Analysis of Journal Policy Effectiveness for Computational Reproducibility"

Minocher, R., Atmaca, S., Bavero, C., McElreath, R., & Beheim, B. (2021). Estimating the reproducibility of social learning research published between 1955 and 2018. Royal Society Open Science, 8(9), 210450.https://doi.org/10.1098/rsos.210450

Allington, D., Brouillete, S., & Golumbia, D. G. (2016). *Neoliberal Tools (and Archives): A Political History of Digital Humanities*. https://lareviewofbooks.org/article/neoliberal-tools-archives-political-history-digital-humanities/

Alnervik, T., Ma, G., & Kohlin, J. (2020). *Success factors in an introductory programming course in a non-CS major* [PhD thesis].

Alvarado, R. C. (2012). *The Digital Humanities Situation* (M. K. Gold, Ed.; pp. 50–55). University of Minnesota Press. https://doi.org/10.5749/minnesota/9780816677948.003.0005

Anderson, K., Bannister, L., Dodd, J., Fong, D., Levy, M., & Seatter, L. (2016). *Student Labour and Training in Digital Humanities*. 16.

Auker, L. A., & Barthelmess, E. L. (2020). Teaching R in the undergraduate ecology classroom: approaches, lessons learned, and recommendations. *Ecosphere*, *11*(4), e03060. https://doi.org/10.1002/ecs2.3060

Balali, S., Steinmacher, I., Annamalai, U., Sarma, A., & Gerosa, M. A. (2018). Newcomers' Barriers. . . Is That All? An Analysis of Mentors' and Newcomers' Barriers in OSS Projects. *Computer Supported Cooperative Work (CSCW)*, *27*(3-6), 679–714. https://doi.org/10.1007/s10606-018-9310-8

Chilana, P. K., Singh, R., & Guo, P. J. (2016). *CHI'16: CHI Conference on Human Factors in Computing Systems*. 1462–1472. https://doi.org/10.1145/2858036.2858323

Connolly, C., Murphy, E., & Moore, S. (2009). Programming Anxiety Amongst Computing Students—A Key in the Retention Debate? *IEEE Transactions on Education*, *52*(1), 52–56. https://doi.org/10.1109/TE.2008.917193

Culina, A., Berg, I. van den, Evans, S., & Sánchez-Tójar, A. (2020). Low availability of code in ecology: A call for urgent action. *PLOS Biology*, *18*(7), e3000763. https://doi.org/10.1371/journal.pbio.3000763

D'Ignazio, C., & Klein, L. F. (2020). *Data feminism*. MIT Press. http://gen.lib.rus.ec/book/index.php?md5=775CC87831CED91A8F2D6B96707987A8

Dawson, J. Q., Allen, M., Campbell, A., & Valair, A. (2018). *SIGCSE '18: The 49th ACM Technical Symposium on Computer Science Education*. 26–31. https://doi.org/10.1145/3159450.3159548

Edelmann, A., Wolff, T., Montagne, D., & Bail, C. A. (2020). Computational social science and sociology. *Annual Review of Sociology*, *46*(1), 61–81. https://doi.org/10.1146/annurev-soc-121919-054621

Edelstein, D., Findlen, P., Ceserani, G., Winterer, C., & Coleman, N. (2017). Historical research in a digital age: Reflections from the mapping the republic of letters project. *The American Historical Review*, *122*(2), 400–424. https://doi.org/10.1093/ahr/122.2.400

Forrester, C., Schwikert, S., Foster, J., & Corwin, L. (2022). Undergraduate R Programming Anxiety in Ecology: Persistent Gender Gaps and Coping Strategies. *CBE—Life Sciences Education*, *21*(2), 21–ar29. https://doi.org/10.1187/cbe.21-05-0133

Forte, A. (2003). *IEEE Symposium on Human Centric Computing Languages and Environments, 2003. 2003*. 285–286. https://doi.org/10.1109/HCC.2003.1260252

Galesic, M., Bruine de Bruin, W., Dalege, J., Feld, S. L., Kreuter, F., Olsson, H., Prelec, D., Stein, D. L., & Does, T. van der. (2021). Human social sensing is an untapped resource for computational social science. *Nature*, *595*(7866), 214–222. https://doi.org/10.1038/s41586-021-03649-2

Greenspan, B. (2019). *The Scandal of Digital Humanities* (M. K. Gold & L. F. Klein, Eds.). University of Minnesota Press. https://doi.org/10.5749/j.ctvg251hk

Heuser, R., & Le-Khac, L. (2011). Learning to Read Data: Bringing out the Humanistic in the Digital Humanities. *Victorian Studies*, *54*(1), 79. https://doi.org/10.2979/victorianstudies.54.1.79

Hockey, S. (2004). *The History of Humanities Computing* (S. Schreibman, R. Siemens,

& J. Unsworth, Eds.; pp. 1–19). Blackwell Publishing Ltd. https://doi.org/10.1002/9780470999875.ch1

Hui Kyong Chun, W., Grusin, R., Jagoda, P., & Raley, R. (2016). *The Dark Side of the Digital Humanities* (M. K. Gold & L. F. Klein, Eds.). University of Minnesota Press. https://doi.org/10.5749/j.ctt1cn6thb

Jackson, L. A., Zhao, Y., Kolenic, A., Fitzgerald, H. E., Harold, R., & Von Eye, A. (2008). Race, Gender, and Information Technology Use: The New Digital Divide. *CyberPsychology & Behavior*, *11*(4), 437–442. https://doi.org/10.1089/cpb.2007.0157

Jacobs, C. T., Gorman, G. J., Rees, H. E., & Craig, L. E. (2016). Experiences With Efficient Methodologies for Teaching Computer Programming to Geoscientists. *Journal of Geoscience Education*, *64*(3), 183–198. https://doi.org/10.5408/15-101.1

Kafura, D., Bart, A. C., & Chowdhury, B. (2015). *Design and preliminary results from a computational thinking course*. 6368. https://doi.org/10.1145/2729094.2742593

Kennedy, K. (2017). A Long-Belated Welcome: Accepting Digital Humanities Methods into Non-DH Classrooms. *Digital Humanities Quarterly (DHQ)*, *11*(3), 17.

Lazer, D. M. J., Pentland, A., Watts, D. J., Aral, S., Contractor, N., Freelon, D., Gonzalez-Bailon, S., King, G., Nelson, A., Salganik, J., Strohmaier, M., Vespignani, A., & Wagner, C. (2020). *Computational social science: Obstacles and opportunities*. 4.

Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabási, A.-L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M., Jebara, T., King, G., Macy, M., Roy, D., & Van Alstyne, M. (2009). Computational social science. *Science*, *323*(5915), 721–723. https://doi.org/10.1126/science.1167742

Locke, B. T. (n.d.). *Digital Humanities Pedagogy as Essential Liberal Education: A Framework for Curriculum Development*. 9.

Manovich, L. (2009). Cultural analytics: Visualizing cultural patterns in the era of "more media". *DOMUS*.

Marsh, A. C. (2013). Omeka in the classroom: The challenges of teaching material culture in a digital world. *Literary and Linguistic Computing*, *28*(2), 279–282. https://doi.org/10.1093/llc/fqs068

Medeiros, R. P., Ramalho, G. L., & Falcao, T. P. (2019). A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. *IEEE Transactions on Education*, *62*(2), 77–90. https://doi.org/10.1109/TE.2018.2864133

Morgan, A. C., Way, S. F., Hoefer, M. J. D., Larremore, D. B., Galesic, M., & Clauset, A. (2021). The unequal impact of parenthood in academia. *Science Advances*, *7*(9), eabd1996. https://doi.org/10.1126/sciadv.abd1996

Nolan, K., & Bergin, S. (2016). *Koli Calling 2016: 16th Koli Calling International Conference on Computing Education Research*. 61–70. https://doi.org/10.1145/2999541.2999557

O'Sullivan, J., Jakacki, D., & Galvin, M. (2015). Programming in the digital humanities. *Digital Scholarship in the Humanities*, *30*(suppl_1), i142–i147. https://doi.org/10.1093/llc/fqv042

Piper, A. (2016). There Will Be Numbers. *Journal of Cultural Analytics*. https://doi.org/10.22148/16.006

Posner, M. (2012). *Some things to think about before you exhort everyone to code*.

https://miriamposner.com/blog/some-things-to-think-about-before-you-exhort-everyone-to-code/

Sharma, R., & Shen, H. (2018). Does Education Culture Influence Factors in Learning Programming: A Comparative Study between Two Universities across Continents. *International Journal of Learning, Teaching and Educational Research*, *17*(2), 1–24. https://doi.org/10.26803/ijlter.17.2.1

skud. (2011). *On being harassed: a little GF history and some current events.* https://geekfeminismdotorg.wordpress.com/2011/10/13/on-being-harassed-a-little-gf-history-and-some-current-events/

Stadtfeld, C., Vörös, A., Elmer, T., Boda, Z., & Raabe, I. J. (2019). Integration in emerging social networks explains academic failure and success. *Proceedings of the National Academy of Sciences*, *116*(3), 792–797. https://doi.org/10.1073/pnas.1811388115

Stodden, V., Seiler, J., & Ma, Z. (2018). An empirical analysis of journal policy effectiveness for computational reproducibility. *Proceedings of the National Academy of Sciences*, *115*(11), 2584–2589. https://doi.org/10.1073/pnas.1708290115

Stroulia, E., Bauer, K., Craig, M., Reid, K., & Wilson, G. (2011). *Proceeding of the 2011 community building workshop.* 20–25. https://doi.org/10.1145/1984665.1984670

Touchon, J. C., & McCoy, M. W. (2016). The mismatch between current statistical practice and doctoral training in ecology. *Ecosphere*, *7*(8), e01394. https://doi.org/10.1002/ecs2.1394

Trinkenreich, B., Wiese, I., Sarma, A., Gerosa, M., & Steinmacher, I. (n.d.). *Women's Participation in Open Source Software: A Survey of the Literature.* 36.

Trisovic, A., Lau, M. K., Pasquier, T., & Crosas, M. (2022). A large-scale study on research code quality and execution. *Scientific Data*, *9*(1), 60. https://doi.org/10.1038/s41597-022-01143-6

Wapman, K. H., Zhang, S., Clauset, A., & Larremore, D. B. (2022). Quantifying hierarchy and dynamics in US faculty hiring and retention. *Nature*, 1–8. https://doi.org/10.1038/s41586-022-05222-x

Way, S. F., Morgan, A. C., Larremore, D. B., & Clauset, A. (2019). Productivity, prominence, and the effects of academic environment. *Proceedings of the National Academy of Sciences of the United States of America*, *116*(22), 10729–10733. https://doi.org/10.1073/pnas.1817431116

Werner, L. L., Hanks, B., & McDowell, C. (2004). Pair-programming helps female computer science students. *Journal on Educational Resources in Computing*, *4*(1), 4. https://doi.org/10.1145/1060071.1060075

Widner, M. (2012). *Learn to Code; Learn Code Culture.* https://www.hastac.org/blogs/michael-widner/2012/02/16/learn-code-learn-code-culture

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215