BLOG@CACM

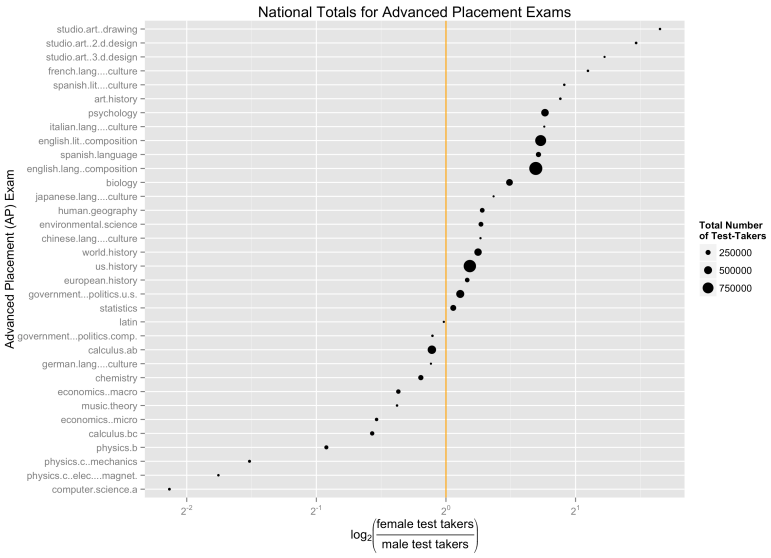
Top 10 Myths about Teaching Computer Science

By Mark Guzdial
July 14, 2015

It's summer time in the Northern Hemisphere. At my university, it's the time when faculty kick back -- and argue about education and teaching on the faculty email lists. Since most faculty arguments about education tend to be filled with more hot air than research and evidence, I'm offering my Top Ten list of Myths About Teaching Computer Science (e.g., all statements that people have said to me) with links to the evidence supporting why I think they're false. If we're going to fight about education, let's bring some evidence to bear.

#10: The lack of women in Computer Science is just like all the other STEM fields.

Not by a long shot. Brian Danielak built this compelling visualization of Advanced Placement exam-takers in 2012. (Advanced Placement examinations are taken by high school students to gain credit or placement in US Universities and Colleges.)



Each dot represents one AP exam. The size of the dot indicates how many students took that exam. Placement in the middle horizontally means that the exam-taking is gender neutral. You'll see most dots are on the right, more female than male. The left most dot, way down at the bottom, is computer science. Here's what's worse: That's a log scale. Sure, Physics is male-dominated, but CS is way more male. CS literally sets the scale for being male-dominated.

#9. To get more women in CS, we need more female CS faculty.

Absolutely, we need more female CS faculty to serve as role models. But the critical factors leading to more retention of women in CS are to create a sense of belonging and to offer encouragement, and those can come from faculty of any gender. We found that in our work in Georgia (see **paper here (http://dl.acm.org/citation.cfm?id=2361276.2361304&coll=DL&dl=ACM&CFID=691874073&CFTOKEN=61621560)**), and it's echoed in the NCWIT report ***Girls in IT***. Joanne Cohoon found that encouragement was the critical factor in getting women to pursue graduate CS (see **publications here (http://people.virginia.edu/%7Ejlc6j/pubs.htm)**), and it didn't matter who did the encouragement. NCWIT offers a workbook for developing a strategic plan for retaining more women (**available here**), and their strategies can be implemented by male or female faculty.

#8. A good CS teacher is a good lecturer.

Active learning strategies are so much more effective than lecturing that an author of recent meta-review of active learning in the *Proceedings of the National Academy of Sciences* says (**in this compelling article in *Wired* (http://www.wired.com/2014/05/empzeal-active-learning/)**):

> *The impact of these data should be like the Surgeon General's report on "Smoking and Health" in 1964--they should put to rest any debate about whether active learning is more effective than lecturing.*

Active learning methods include peer instruction (here's a great website for **peer instruction in CS (http://www.peerinstruction4cs.org/)**) and peer-led team learning (see **the Center of Peer-led Team Learning here**). The first article that really caught the attention of even the skeptics was Richard Hake's 6000-student 1998 study showing that active engagement pedagogies led to more physics learning. The PNAS article (**linked here (http://www.pnas.org/content/111/23/8410.full.pdf)**) is the latest and most definitive argument for active learning strategies, surveying over 158 individual studies.

A good CS teacher should use the best methods available. Lecturing is not the best teaching method available. There are lots of better methods, and the evidence suggests that teaching with these better methods would lead to better learning (see **blog post here**).

#7. Clickers and the like are an add-on for a good teacher

No, active learning methods with even a novice teacher are more effective than a good, experienced teacher. In a study in *Science* in 2011 (see **summary paper here (http://news.sciencemag.org/2011/05/better-way-teach)**), Nobel laureate Carl Wieman taught a post-doc and a graduate student some active learning methods, then had them teach one week of a section of Physics while a tenured physics professor taught the comparison section. The post-doc and grad student had better learning and better student motivation (e.g., more students came to lectures) than the one taught by the physics professor. No matter how you cut it, active learning wins out over lecture, and *everyone* should be using them.

#6. Student evaluations are the best way to evaluate teaching.

If what you want to know what students *like*, so that they become happy and donating alumni, then student opinions matter. If you want to know what's *effective in teaching*, there are better options. Carl Wieman is promoting a method where we judge professors based on an inventory of their teaching practices (see **article in *Change* magazine (http://www.changemag.org/Archives/Back%20Issues/2015/January-February%202015/better-way-full.html)**). Phil Sadler found that the best teachers can accurately predict what their students will get *wrong* on tests (see **article on his paper (http://news.harvard.edu/gazette/story/2013/04/understanding-student-weaknesses/)**), because teachers who know the common errors know how to address them. The Gates Foundation did a three year study to develop more effective teaching measures (see **press release here (http://www.gatesfoundation.org/media-center/press-releases/2013/01/measures-of-effective-teaching-project-releases-final-research-report)**). There are far better methods for evaluating teaching than just asking students if they thought the teacher was entertaining.

#5. Good teachers personalize education for students' learning styles.

The best research evidence is that there is no such thing as learning styles. There is no good evidence that some students are visual learners or auditory learners. The NYTimes article in 2010 (see **link here (http://www.nytimes.com/2010/09/07/health/views/07mind.html?_r=4&hp=&pagewanted=all)**) was where I first learned about the evidence, but now there are also books by Kirschner and his colleagues (see **Amazon link here (http://www.amazon.com/Urban-Myths-about-Learning-Education/dp/0128015373/ref=sr_1_1?s=books&ie=UTF8&qid=1436553396&sr=1-1&keywords=urban+myths&pebp=1436553395968&perid=0EK061PQN6TDECDQQYH3)**) and Lilienfeld and his colleagues (see **Amazon link here (http://www.amazon.com/Great-Myths-Popular-Psychology-Misconceptions/dp/1405131128/ref=sr_1_3?s=books&ie=UTF8&qid=1436553447&sr=1-3&keywords=lilienfeld)**) debunking the "urban myth" of learning styles. There is evidence that teachers who teach to learning styles are actually harming their students' learning (see **NPR piece (http://www.npr.org/sections/health-shots/2011/08/29/139973743/think-youre-an-auditory-or-visual-learner-scientists-say-its-unlikely)**), because those teachers try to provide those students predominantly with instruction in one style or modality, when it's better to teach with a vareity. The myth is surprising persistent, with a MOOC recently offered on teaching to learning styles (see **review here (http://www.davidjoyner.net/blog/i-cant-say-anything-good-about-most-moocs/)**).

#4. High schools just can't teach CS well, so they shouldn't do it at all.

There are too few computer science teachers in the United States (see **report from U. Chicago (http://outlier.uchicago.edu/computerscience/OS4CS/)**), and around the world (e.g., **see a recent report from Scotland**). But there are some amazing teachers out there who compare very favorably to the best teachers in math and science. We can use student performance on the AP exams (which the College Board works hard to make roughly equivalent in difficulty across subjects) as one metric. Here's a statistic that AP CS teachers have been talking about in social media. Worldwide in 2015, there was 1 perfect AP Calculus Level BC (harder one) exam (i.e., one student in the whole world got every question right on the Calc BC exam), 2 perfect AP Statistics exams, and 3 perfect AP Calculus Level AB exams this year. There were 66 perfect AP CS Level A exams.

#3. The real problem is to get more CS curriculum out into the hands of teachers.

The *amount* of curriculum is not the right way to think about this. There are *SO MANY* CS curriculum repositories now with so much curriculum. Google and NCWIT just set up **EngageCSEdu** to promote materials that are highly engaging. Maybe you just want to use open source materials. You might be interested in the Computer Science Open Educational Resources Portal (CS OER Portal) (**http://iiscs.wssu.edu/drupal/csoer (http://iiscs.wssu.edu/drupal/csoer)**). For high school teachers, there is the **CSTA Source Web Repository**

(http://csta.acm.org/WebRepository/WebRepository.html) and the Resources section of the **Computing at Schools website** (http://www.computingatschool.org.uk/) (which is pretty big and growing almost daily). Specifically for a particular tool or approach, there's the **Greenfoot Greenroom** (http://greenroom.greenfoot.org/door) ,**ScratchEd for Scratch Teachers and other Educators** (http://scratched.media.mit.edu/) , **Bootstrap World** (http://www.bootstrapworld.org/materials/fall2015/index.shtml) , the **Alice teacher's site** (http://www.aliceprogramming.net/) , the**TeaParty site** (http://home.cc.gatech.edu/teaparty) (for the Alice + MediaComp book), and of course, the **Media Computation site** (http://mediacomputation.org/) . There are many others - for particular books (like **this one introducing Python with Objects** (http://www.pearsonhighered.com/educator/product/ObjectOriented-Programming-in-Python/9780136150312.page#dw_resources)), and for particular teaching approaches (like**Exploring Computer Science** (http://www.exploringcs.org/) and **CSUnplugged** (http://csunplugged.org/)).

Taking what you do in your University classroom and just putting it in one of these repositories is unlikely to meet the needs of high school CS teachers.

We need CS curriculum materials that meet what teachers *want* and *need*. We need research about what CS teachers want, when they'll use repositories, *and* when they won't. We very much need *targeted* CS curriculum. The new AP Computer Science Principles class is being rolled out nationally (see **site here** (http://apcsprinciples.org/)). High school CS teachers could certainly use curriculum materials for that course because it's new. But "*more*" CS curriculum, without qualification or just thrown over the University wall, is unlikely to be of much help.

#2. All I need to do to be a good CS teacher is model good software development practice, because my job is to produce excellent software engineers.

Increasingly, the students in your CS classes probably don't want to be software developers. That's a good thing because the Bureau of Labor Statistics says that we don't need many more software developers (see **report here**). Eric Roberts saw back in 2011 (see **his post here**) that the rise in enrollment in CS is mostly coming from students who want CS *plus* something else. They don't want a job programming. They want to know programming *to make themselves more marketable and effective in some other job*. Scaffidi, Shaw, and Myers predicted this back in 2005 (see **blog post on their story here**). For every professional software developer in the world, there are *nine* more people programming as part of their jobs who aren't professional software developers. While it may be that your department may define your job that way, but for most CS teachers, it's likely that most of your students are *not* aiming to ever be professional programmers.

And the number one biggest myth about teaching computer science:

#1. Some people are just born to program.

There is no evidence-based reason to believe in a *Geek Gene*, the idea that some people are just *born* to program and those lucky few have a "Geek Gene." First, it is far more likely to believe that differences in ability that we see in computer science classes are due to experience and learning, not genetics (see **argument here**). Second, there are reasons to believe that better teaching can trump even differences that might be innate, like the difference between women and men in spatial reasoning (see **argument here**).

The big problem is that the Geek Gene is not only wrong, but *belief* in the "innate differences" is likely contributing to the lack of diversity in computing. A recent paper in *Science* found that belief in innate differences (like the "Geek Gene") is a likely cause for too few women in a field (see **article in *Science* discussed here**). If you think that some students just aren't "wired" to program, you as a teacher are unlikely to do as much to help them learn.

I suspect that our belief in the Geek Gene stems from too little evidence about effective teaching and learning (see **blog post here**). If you don't know what's working and not working, it's easy to believe that our teaching isn't making much difference and that the best students are just "*born*" that way. That's the easy way out for CS teachers, and does our students a disservice.

Thanks to Briana Morrison, Mikala Streeter, and Barbara Ericson for thoughtful comments on previous versions of this post.
