The N=1 guide to grad school (and hopefully, knowledge work)

by Adam Marcus, with friends

[hide navigation]
- Introduction
- Caveats
- Approaching things
- Creating things
- Putting yourself out there
- Other people
- Staying sane
- Peer review
- Thank you

Reach out on Twitter.

Introduction

Last August, I finished my Ph.D. with the help of a bunch of amazing people. I submitted my thesis, and on a bus in Turkey during my post-graduation vacation, I put together a list of practices and mindsets that I sometimes stuck to while working on my Ph.D. Over the last few months, I added some details, and you'll find the finished work below.

I promised myself I'd never write a self-help guide, but as I went through grad school, I found myself repeating the same advice to newer students, and decided to write it somewhere. I hope my suggestions take you down a healthier path through your Ph.D. And I hope, for my own sake now that I'm at Locu, that the advice applies to other similarly stressful knowledge work jobs.

Got something to say about this document? Reach out to me or fork the guide and make your own.

Caveats

There are a boatload of caveats that limit the effectiveness of my advice. First and foremost, my advice is based on a sample size of one (thus, it's an N=1 guide). The suggestions here are things that appear to have worked for me and perhaps a few other people, so you're not getting advice from a very large sample.

Additionally, this advice applies to Ph.D. students, perhaps only in Computer Science, and perhaps only if they build systems. Systems is a subfield of Computer Science that is experimental and engineering-heavy. In systems (e.g., database systems, operating systems, networking, some parts of human-computer interaction), you optimize for novel engineering efforts and don't offer as many rigorous proofs that your system is better than another. The best Computer Science systems researchers put out somewhere between one and four publications a year, and it's often rare that they work on the same topic during their entire Ph.D. This makes a Computer Science systems Ph.D. different than, say, a Physics or Math one where a student works on far fewer topics and writes far fewer publications while in school.

Since I finished up recently and sometimes practiced these techniques, I'd like to believe they helped me succeed. I might, however, be suffering from something called confirmation bias, so feel free to call me on advice that seems downright out-of-place.

Finally, I had it good: my advisors were all extremely supportive and helpful in different ways, and my funding situation was such that I didn't have to worry too much about where my stipend was coming from each year. I can't imagine having gone through grad school without the ever-present and diverse types of support that my advisors gave me, or with the constant fear of not receiving a paycheck next month.

You can be successful if you blow these suggestions off. Your challenges might be very different from my own. I have seen many happy systems-building Computer Science Ph.D. students succeed without applying the ideas below, so they are certainly not a requirement even in my subfield. Shape your own path, and feel free to borrow any of the ideas below to get there.

Approaching things

You are wonderful

You won't hear this enough. You won't say this to people enough. You are wonderful. You are wonderful. You are wonderful.

Be positive (about your own work and that of others)

While it's the most cliche-sounding advice, it might be the most important. You're a scientist, so it's your job to be skeptical of others' findings, and hold them to often-difficult tests of correctness. But you're also a human who works with other humans, all of whom have feelings.

There are three places where I think the community, and in particular graduate students, could benefit from being more positive:

- **Be positive about your own work.** This might be the hardest. As the person who spends all day thinking about your own research, you are the one who knows where the holes are. But you're not researching holes! When someone asks you about your work, it's fine to provide caveats, but don't lead with your limitations! (Except when writing self-help guides :).) What are three wonderful and novel things about your work? Why does it stand to make some small part of the world better? If you can't get excited about three aspects of your work, it's possible you should consider a topic you think matters more.

- **Be positive about your colleagues' work.** There will be many moments in your grad student life where a fellow researcher comes to you with a new idea, or is trying to hash out the arguments they are going to make in a paper. You do them no service by being a Yes-Person: if this exact idea has been done before, or if the arguments hold no water, save your friend the time of finding this out in peer review. That said, it's almost never the case that an idea has been implemented in quite the way your friend is describing, and it's even less likely that the particular arguments are so terrible that your friend should move on with their life completely. What is the coolest aspect of their work? What is the one grain of an idea that, if it made its way out of research-land, would make some part of the world that much more awesome? Help your friend find this grain, and suggest a next step in implementation or writing that will optimize for that awesomeness. Your friend came to you in what is likely a vulnerable state after avoiding enough self-criticism to share their idea. The least you can do is be excited about one or two directions to go from here.

- **Be positive when reviewing a paper.** You've likely been in this situation: You were asked to present someone else's paper at a reading group. You sink the time into reading the paper and preparing a presentation about it. You then spend an hour of your and everyone else's time deconstructing the paper, pointing out all of its failings. In doing so, you have wasted everyone's time. No one will

> remember just how the paper failed to impress you, and they walk away from the meeting without any new thoughts about how the world works. You should aim to start such discussions with a small list of interesting things you learned from the paper. Maybe the system sucked, but the experiment design was cool. Maybe the experiments proved nothing, but the implementation details were particularly interesting. Find something novel that the paper proposed, and make sure that people walk away with that new thought. The same principle applies to reviewing papers in submission. The authors just spent months or years on the work they are submitting, and they aren't stupid. Don't assume the paper should be accepted (most aren't), but don't sit down to read the paper looking for reasons to reject it. That's a depressing way for you and the authors to spend your time.

You're in charge

One of the things it took me too long to appreciate in grad school was that the only person who can decide what to do next is you. Your advisors are smart collaborators with lots of experience, but they aren't your boss, and have no training in management. They're also a little too busy to manage what you work on, who you work with, and how you spread the word about your awesome findings. That's on you. It takes a while to appreciate this, but once you do, it's actually quite freeing.

Early on in grad school, you can go to an advisor and ask them what exciting projects they can think up. They will likely list a few that you can take up and dig into. They can meet with you regularly, advise you that you've gone off course or tell you that of the three choices ahead of you, the second one seems most promising. If you're lucky, they might even lay down some heavy groundwork for a paper or edit the hell out of your writing when they have a better way to deliver the message. But it will ultimately be on you to decide what to work on, who to work with aside from your advisors, what the exciting next steps are, and how to tell the world that your research is interesting.

As you get into the middle of your grad school career, start to branch out and work with new people. Start writing more of the introduction to your papers. Start testing the waters in blogging and telling people what it is that excites you at conferences.

It's unlikely that you will leave grad school with no dependence on your advisors. At the very least, they will be the ones who remind you how underdeveloped the ideas in your thesis are, and give you advice on where to go from here. And they will (especially if you take the academic career path) be the people who introduce you to many interesting job opportunities and collaborators. But I think it helps to think of your advisors as smart people who want you to succeed, whose advice is useful when you ask the right questions, whose connections are useful when you know which connections you want, and whose feedback is most useful when you come to them with something concrete to get feedback on.

Impostor Syndrome can die early, but you can still feel like a hack

In any group of high-achieving people, a discussion of Impostor Syndrome will come up. That's fair: you may very well feel like you don't deserve to end up where you did given the awesome people around you. But if you're healthy about these things, you will eventually stop trying to compare yourself against your very different colleagues, each with their very unique set of properties that make them awesome.

Sadly, even once you stop using other people to gauge your own success, you may still feel like crap.

It will likely come from within. It will also likely come near a major deadline. You won't feel inadeqeuate because your colleagues are smarter and more deserving than you: you will feel bad because you think your own work simply doesn't have a leg to stand on.

As the person who has spent the most time with your own work, you will be the most critical of it. You will know all of your work's weak points. You will find ways to discount all of the meaningful supporting arguments that you thought up in the introduction to the paper you're working on.

The most important thing to do at this point is to be objective. Is your algorithm just plain incorrect, or is it just the case that it isn't optimal in all scenarios? Do your experiments not support your claims at all, or do you just have to make less strong claims than you initially thought?

This is where advisors and other grad students should come in. There may be really good reasons to delay submission of your work in its current state. Don't waste reviewers' time with work that is obviously flawed. But after several weeks of internal stress, bad sleep, and playing the role of the skeptical scientist, you might be a bad judge of the overall value of your work in the days before a deadline. Bring in someone with less at stake, ask them to be honest, and if they are even slightly positive about your work, ignore your self-doubt and push the submit button.

Creating things

Crappy first drafts

Let me pause advice mode and go into story-telling mode for a moment.

- **Story 1.** If you're reading this, there's a high probability you don't need me to introduce you to Michael Bernstein. I was lucky enough to work with Michael on a few projects, and shared an office and a few meals with him. One day we were at lunch (at Steam Cafe!) discussing what would eventually become the seeds for our theses, and he told me about his next project idea. Michael had an idea around "embedding crowds in user interfaces," and I walked away figuring he'd find a new topic once he realized that the topic wasn't concrete enough to do anything with. The next time I heard of the topic was about a month later. Michael had written a high-fidelity introduction to a paper. He hadn't yet built a system or run experiments, but he had 1.5 pages of concrete ideas written out. I read it. I couldn't stop thinking about it. I wanted his idea to exist! It was wonderful, and anything but abstract. It also ended up being the first version of Michael's Soylent paper, which, if you've done work in the area of practical crowd-powered systems, you know to be one of the most influential in the field. By the time the research was done and the paper was submitted, the introduction looked a bit different than Michael's first version. But the first draft was enough to get a feel for how awesome his idea was.

- **Story 2.** Ted Benson is a self-described "engineer's engineer." He can implement an end-to-end prototype almost as fast as I can describe a system. This is one of his many strengths. The DataPress paper we ended up writing was about a different system than the first version he built in a weekend, but the version he built in a weekend was part of the inspiration to keep working on the project. Because of his prototype, we had something tangible to look at, reflect on, and improve in version 2. Ted's ability to rapidly prototype ideas and not hold them up as holy when we need to scrap certain failed assumptions shaped how I approached future design work, and I think it was for the better.

Stop being a perfectionist, especially when working on version one. It is important that you finish your first draft, however crappy you might think it is.

Your first draft of a paper or a prototype (or a paper prototype!) gives your advisors and collaborators something concrete to give you feedback on. The people you're getting feedback from are likely busy, and catching them up on abstract design concepts without something concrete to point to makes for equally abstract and unactionable advice.

I find it helpful to tell myself "I'm going to sit down and write something crappy. That's fine, because I'll edit it if there is time, and I have co-authors and advisors who won't let me submit a final version that's crappy." This mental mode allows me to stick to write-only mode for my first version of something, and allows me to avoid getting paralyzed with self-criticality. To avoid getting distracted by things I can't remember or need to look up, I insert "XXX" or "$$$" wherever I need to revisit the text after I exit write-only mode (thanks, Sam!).

In part, the "crappy first draft" mantra works because everyone has a little bit of perfectionist inside of them. It's hardly ever the case that I submit my crappy first draft to my advisors or friends without giving it a read-through and editing pass. In that respect, "crappy first draft" is a good way to separate the writer in you from the editor in you.

While you're writing or building, you might feel like your idea sucks and want to stop it early. Unless you're totally convinced after some more thought that your idea is unworkable or totally impractical, keep going. That feeling of discomfort you have might turn out to be your core research problem, and you'll feel better once you finish a draft and identify the unsolved work ahead of you. Even if you're not convinced your idea is great, you should continue. When you present the introduction to your advisor, they might think of something completely different and more exciting. Your crappy first draft will end up being a wonderful first seed for the idea you'll actually start working on!

There's a flipside to all of this. While it's often the case that you improve on your first draft, you sometimes run out of time. This is especially the case for research-grade software. When Ted Benson and I worked on the SyncKit paper, I spent two days of the last week of our paper deadline trying to write ever-more-realistic browser traffic workload generation models for our experiments. Ted knew we had more important work to do. For example, we didn't have a system or API description in our paper. He was pretty direct in reminding me that a perfect model and no paper was less good than a simple model and a completed paper. I implemented a simple (unrealistic! :)) model, ran the experiments, and moved on to writing. We never went back to make a more realistic model: SyncKit's design sections were far more critical to understanding why it was neat, and the experiments turned out to be more than sufficient for anyone with a realistic model of the web in their mind. The paper got in, the first draft of the workload stayed crappy, and the world of web performance was no worse off for it.

Make something cool

One of the most wonderful pieces of advice that I've gotten was from one of my advisors, Sam Madden. I won't speak for his grand vision of life, but several times he's told me to just "make something cool." This advice has two components: make, and cool.

- **Make.** Stop talking about a wonderful idea you have, and prototype it. People's experiences in interacting with a tool are more valuable than their experiences with what they think you are saying.
- **Cool.** Not deep. Not intellectual. Not novel. Not fundamental. Just cool. This is not the only way to do research, but it was the only way I managed to work on research I cared about. Cool prototypes help you feel good about your contribution. They also help you write a convincing introduction to a paper with certainty that at least one tool can benefit from the science you're about to share. Coolness is not the full story: you better find some novel kernel in that cool thing you made, or else you won't have much to share with your community.

This is, admittedly, very systems/engineering-oriented advice. It also helps to figure out what your research community thinks is cool. One way to do this is to meet folks at conferences and workshops early and often, as I discuss below.

Make cool things. Sometimes they will contain a nugget of interesting research.

Establish quiet time, and make it work time

Later on, we'll talk about how frequently you should take breaks from work. When you do sit down to work, it's important to spend that time meaningfully.

This advice is less important for the two weeks before a deadline, where external pressure is usually enough inspiration to get things done. It's important to make forward progress in the months between deadlines, and it's important to do so explicitly.

Schedule two to three hours of low-interruption time every day you plan to work. Block off time on a calendar if you have to. Find a spot that works for you: a library or a coffee shop are reasonable places, depending on how much background noise you like to have while working. Your office is usually a bad place to work during this time, as it's full of fun people to talk to instead of getting work done.

During those hours, do work. Don't answer emails. Turn off wifi if you need to. Practice the "crappy first drafts" mantra above to avoid doing too much web searching that will cause the hours to melt away on Wikipedia. This is not time to read papers: it's time to make something, or write about something you've made!

Two to three hours a day of productive creative work might strike some folks as a small amount of time to work each day. It's really not. If you write one page per hour, a year of 2.5-hour days with weekends off and an obscene 60 days of vacation can result in a 500-page Ph.D. thesis! Surely in six years, you can accomplish something good on that schedule.

And it's not that you're not working the rest of your day: spend time in the office to meet with people, discuss new ideas, publicize your work, and explore new topics. One exception to the two-to-three-hour creative workday is coding sessions that sometimes require more ramp-up/cool-down time. Remember: I'm not telling you to ONLY do two to three hours of creative work a day, but make sure to schedule at least that much time each day to be productive and avoid distractions.

Today doesn't matter

Given the math we just did about how much you can get done if you focus on making headway for a few hours a day, it's also important to balance this knowledge with the fact that some days (or weeks!) you just won't produce much of substance. Some days it's a concentration issue, while others are filled with work that you thought was important, but turned out not to be. It's also likely that over the course of your years in grad school, some external life event will happen that makes it impossible to be productive.

Being unproductive as a knowledge worker is frustrating enough. Your job on unproductive days is to, if at all possible, try to remove the barriers that stopped you from getting something done, while not beating yourself up too much for the lack of productivity. Remind yourself that today likely doesn't matter on the cosmic scale of grad school, and focus on tomorrow.

Don't get complacent. Don't yak shave. Don't repeat "today doesn't matter" every day. Make sure to keep people in the loop when you think you're going to miss a deadline. But when you hit a few days of lost productivity, beating yourself up will likely just result in a few more days of the same.

Read papers, but not too many, and not to a state of paralysis

Reading other people's papers is a blast, particularly when they are good. It's important to keep track of what your field is up to, and it's mind-widening to keep up with research a little bit outside of your realm of expertise. Still, it's possible to overdo paper-reading.

Read papers as a way to open your eyes, rather than as a way to discount your ideas. You shouldn't walk away from a paper-reading session with the idea that all of the good ideas have been solved. Should you find yourself thinking this way, stop reading for a while, and start making something. Once you've put some time into making it, you will likely have an interesting thing to say about it!

Inside most interesting research is an unanswered question. What's yours? It's rare that someone has already solved the exact problem you've solved in the exact context you're solving it. And even if they have, it's likely that five to ten years of technology advancements give you something interesting to say about the problem space regardless.

Talk about many things, work on a few at a time, and commit to finishing most things

After working on nothing but my Master's thesis topic for two years, I was burnt-out and couldn't bear to touch the topic again. To avoid such burnout, I spent the two years after my Master's thesis overcompensating by working on three to four projects at a time. This resulted in a bunch of fun but less-than-deep experiences in doing research. My advisors and at least one collaborator took note and helped me realize I should pare things back a bit.

To say the least, working on too many or too few projects at a time is detrimental to getting things done. What worked best for me was to actively contribute to about two projects at a time. This allowed me to jump between projects if I got stuck, but didn't pull me in so many directions that I got nothing done.

If you need more mental stimulation than two projects at a time can provide, try to talk to people about what they are doing, or have deeper conversations with someone about the next project the two of you can work on. It can sometimes take a few months of chatting about an idea before you jump to solving it with someone, and pipelining the ideation process is a nice way to provide you with novel ideas without encroaching on your current projects.

It sometimes helps to make your active projects relatively different from one-another to suit your differing moods. Having a writing-heavy project and a coding-heavy project helps you have something to do if you wake up one day and feel like doing one over the other.

One last note: commit to finishing as many of the things you start as possible. This doesn't mean you need to write a paper on each inkling of an idea you discuss with a friend. But when working on multiple projects at once while also talking to collaborators about exciting future projects, it's possible to never take existing projects to completion. I've seen this shiny-new-project phenomenon happen with plenty of grad students, myself included. It makes sense that a new project is more exciting than the last 20% of an old one. Still, it's good to commit to finishing something useful (a paper, an open source project, etc.) before moving on to the next thing. Academia isn't just about scratching interesting itches: it's also about sharing the results of those itches with the world!

Have side projects

(I'm hesitant to suggest this after telling you to limit the amount of projects you work on concurrently, but grad school takes a few years, and it's OK to be inconsistent across years.)

Try to work on something creative outside of the world of research. If you can, work on a project that you own in some way but are never required to answer "Why is this novel?" or "How rigorous was your

procedure?"

During grad school, I worked on a startup with some friends every saturday for two years, took some classes about poverty and development, wrote a book chapter on a topic outside of my research area, taught high-schoolers in Jerusalem, and taught a few not-for-credit courses on programming and data. I'm not saying these things so that you think I'm some sort of super-acheiver. Taking these diversions from research came at the expense of more focused research. But the diversions also helped me interact with different people, think about different problems, experience different stresses, and have an outlet outside of research when I needed a break. Some of them were also more interesting to talk about with a random person I met than the particulars of my research at any moment, so there's also that!

Putting yourself out there

Self-promote

I used to view self-promotion as a bad thing. It's still hard for me to do, because talking about yourself when so many other important things are going on in the world seems a bit myopic. But unlike working at a company, where there are public relations and marketing folks whose job it is to make sure people know about you, that job in grad school is in your own hands.

It turns out that most universities have their own news offices, and the job of those offices is to spread word of the cool things going on across campus. Speak to someone in that office. It might turn out they wanted to learn more about your topic and help you share it with the world. That would be nice!

I'm not telling you to ignore what other people say and blabber at the world about how cool you are. I'm telling you to feel about as confident in asking someone what they do as you feel in telling them that you're working on something interesting when asked. If you're not totally confident doing it in person, do it on Twitter. I'm positive someone is interested in what you're doing.

Part of being a researcher is being a communicator. You owe it to your lab to get other researchers excited to work there. You owe it to the funding agencies that pay your salary to tell the world why the research you do matters. You owe it to society and to the researchers of tomorrow to get people excited about research. And you owe it to yourself to tell people that what you do is pretty neat, which it is.

Keep a blog, take notes

Early on in your research career, before you develop your own opinions and intuition as to what makes for good research, you spend a lot of time learning the language of your field, and practicing understanding problems in a principled way. This is not something folks outside of academia spend as much time doing, and gives you a unique set of skills to share with the world if used properly. (You can also end up sounding disconnected, demeaning, and overly academic, and you should actively avoid this.)

A blog is a place for you to learn to write in a way that's not overly academic or scientific, and does not require peer review to post. Like trying your hand at research, however, writing your first few posts might feel overwhelming. *What novel piece of text can you share with the world? What if you aren't very good, and no one likes what you have to write?* If you're crippled by these sorts of worries, there's still hope! Rather than coming up with your own topics to cover, you can use your ability to understand and explain difficult topics to share notes you've taken with the world.

Some of my most widely distributed blog posts are collections of notes that I took in the course of being a graduate student. Two examples are particularly memorable. As a teaching assistant in MIT's database

course, I had to attend every lecture, but had already taken the undergrad and graduate versions of the course. I took advantage of knowing the content relatively well to try my own hand at taking notes that students in the class could use, and then released them for a wider audience on my blog. I still hear from students who appreciate my putting the materials online, which is amazing considering I had nothing to do with generating the course content I took notes on. Another fun example comes from when I attended and took notes on a Boston NoSQL conference. The notes made their way onto places like HackerNews and Reddit, and two days later I was contacted by Greg Wilson to write a book chapter on NoSQL for a book he was editing.

It's cool that my two most useful blog posts were simply note-formatted summaries of things that other people are responsible for. One turned out to be a good resource for other students, and another allowed me to try my hand at writing a book chapter. Not bad for what is essentially a side effect of being a grad student!

Start the conference circuit early

I didn't go to a conference until the end of my third year in grad school, which was a huge mistake. When I finally did start going to conferences, I realized that many peers who went since their first year built relationships with students and professors at other schools. Most of the folks you meet at conferences will be the folks you will work with and speak with if you stay in an academic setting. They are the people who will champion you to their department when you're looking for a job, and they are the people who will collaborate with you on interesting projects you chat about over dinner. Starting the conference circuit on year one means you meet more people, get more advice, and have more vociferous champions when it's time to get on the job market.

One of the things I realized too late in grad school was that you don't need to have a paper in a conference to go to it. Most professors with stable funding sources that I've interacted with will fund your trip to any conference you're presenting at, and at least one conference a year even if you aren't presenting. Take them up on that. Don't feel guilty, especially early in your career, to attend conferences just to meet people. There are even ways to make the experience more worthwhile for everyone. If you're not ready to submit a paper, submit a poster or demo. These won't make your career, but they will allow you to attend the conference with something to talk about and get feedback on, so you don't feel as guilty for just showing up. Some conferences also invite student volunteers to help out with administrative duties in exchange for covering some degree of room/board/conference fees, and are a nice way to meet your peers at other schools.

And once you reach the conference, don't spend your entire time going to talks. Paying attention to deep technical content is draining, and you only have so much attention. You should pick some number of topics (likely less than 10) that you actually want to learn more about, and concentrate on those topics in earnest. Additionally, maximizing talk-watching time minimizes meeting and conversing time, which, while also tiring, is crucial to broadening the set of folks you will be excited to meet and work with in the future. You will notice successful professors attending some talks and sitting in the halls interacting with colleagues the rest of the time. Learn from their experience.

Understand the value of workshops

Early on, I misunderstood something important about academic workshops. I was told that workshop papers are not as prestigious as conference papers. To get a good research job, I should focus on getting papers into prestigious conferences. These are both true statements, but miss the point of attending a workshop.

A workshop is a small gathering, sometimes attached to a much larger conference, where like-minded people go to network. Some examples are a crowdsourcing workshop at a human-computer interaction conference, or a social network data workshop at a databases conference.

People don't go to workshops because they are prestigious, or because their career will be made by writing a 2-page position paper that serves as proof that they really want to participate in the workshop. They go to meet people outside of their department who share a common interest. They go to hear what everyone else is working on. They go because sometimes, an idea that they start at the workshop ends up being the paper that's described as the most thought-provoking one at a conference the next year.

Workshops are important because they remove a socially awkward feature of large social gatherings: at a workshop, you *know* that the person you're talking to is genuinely interested in speaking with you about the workshop's topic, and that the two of you have a *shared interest*. You know that they are likely to want to collaborate with you on a topic that you're just jumping into. Finally, workshops are a good forum to start a conversation with a research hero who would love to speak with you but would be too exasperated by the flood of people at a larger conference.

Don't go to workshops for the prestige. Go to a workshop because it's a good place to meet people, share ideas, and start new relationships.

Teaching != teaching assistantship

Most Ph.D. programs have a teaching assistantship requirement. Usually this means you team up with a professor and help them with their class.

But teaching assistantships aren't about teaching per se. Most of your work as a TA will be in grading, writing some homework and test questions, and holding office hours. In introductory courses, you might get some face time with students in recitation, but this is rare for advanced courses.

In short, TAships aren't designed to teach you to teach. If you're at all curious about whether you love to teach, participate in programs that allow you to do curriculum development and treat you like an instructor.

Another secret: no sane department will argue if you tell them you want to teach a class on a topic of interest. MIT has a wonderful monthlong period in January where anyone can teach courses like an Intro to Java or Data Literacy, and other schools offer similar programs. If you want to know whether you enjoy teaching or not, try your hand at teaching a short course.

Other people

Work with someone

Aside from a stint in working with Dan Abadi and Kate Hollenbach in my first year of grad school, my first two years in research were solitary. I had my weekly meetings with my advisors and the rare two-minute research pitch I'd give someone, but I otherwise didn't collaborate with others on what became my Master's thesis.

While there are lots of good graduate students who work solo, getting feedback from my advisors once a week was not enough interaction for me to get good work done. After finishing my Master's thesis and being so burnt out on it that I couldn't bring myself to successfully publish it, I promised myself I would only work on projects with other people. With that change, I was now talking to someone about the deep aspects of research at least once a day. I had a partner in architecting and building the nitty-gritty of the various systems we were working on. When one of us would get down about the state of a paper or the featureset of a prototype, the other would still be hacking away, pushing toward a deadline. When I worked with other people, we would set a deadline and submit a paper for that deadline. I was accountable in a way that weekly advisor meetings could not facilitate.

One particularly poignant example of this is Qurk, the system that ended up forming the basis for my Ph.D. thesis. I had thought about Qurk quite a bit, and had even written a thesis proposal that contained the key ingredients for future work. But it wasn't until a week before a conference that I mentioned submitting a short paper on Qurk to Sam, my advisor, and Eugene Wu, my friend and collaborator. I threw the thesis proposal into conference paper format, edited it to be slightly more paper-like, and sent the crappy first draft their way. They were nice enough to not tell me it was crap, but provided many constructive comments and edits that (along with more edits by David and Rob, my other advisors) made the final version what it was. This paper, while low on details, provided us the basis from which to work on a few more detailed followup papers. Had I not invited Eugene and my advisors to spruce things up, I likely wouldn't have had a paper of the form we ended up with, and certainly not within a week.

There's a caveat to all of this collaboration that didn't end up affecting me because I haven't participated in an academic job search. When you co-author work with others (depending on your field), you will no longer be first author on all of your papers. As you ask about your academic job prospects, you might hear from people "do you have enough first-author papers to get a good job?" or "I can't really determine from your publication list how many of these projects you led." To the extent that their concerns are legitimate, you should ask yourself if you want to change how you work to support a system that's still susceptible to the myth of the lone genius. And besides: if working alone makes you unhappy and results in less publications, it's better to work hard with a collaborator you are successful at completing work with and sharing the glory of grad school.

Learn to separate other people's excitement for your own

Early on in undergrad and grad school, I confused two things: 1) being excited by other people's passion for a topic, and 2) being excited by a topic. Speaking with someone who is legitimately excited about a topic is a good way to feel good, and you should do it frequently to open your mind and to learn about new things. But before committing yourself to a project, step away from someone else's excitement for it, and ask yourself whether the topic matters to you. It's possible to find yourself in the middle of three projects that other people care deeply about, but you just joined onto because of someone else's excitement.

Open up to others

Opening up to others is one of the best ways to realize that many of your fears and stresses are not unique to you. Rather than harping on what frustrates you, sharing your thoughts with other grad students and people outside of grad school can help. If you're lucky enough to have a life partner while in school, talk to them about your day, and sort one-another's thoughts when they become tangled.

There is a limit to this advice: sometimes, the best way out of an uncomfortable situation is to finish what you started. Speaking with others can give you relief and a common sense of camraderie, but speaking to anyone who will listen will eventually just hinder you from finishing the tough task you need to perform. Talk through things that bother you, but make sure you're actively working to resolve them as well.

Staying sane

Say 'I'm overwhelmed' when you are

It's your advisor's job to push you toward various forms of progress. Often this means pushing you when you're slowing down to make sure you hit a deadline. There are cases, however, when it doesn't make sense to push you as much as it makes sense to give you guidance because you're overwhelmed. Use the phrase sparingly, but become comfortable with saying "I'm overwhelmed" at the right moments.

Being overwhelmed isn't an escape hatch: you can't always stop working to make things better. But when used at the right moment, it can help start a conversation for planning and course-correcting a project before it's too late. Ideally, you will tell your advisor or collaborator that you're overwhelmed two weeks before a deadline rather than giving them two days of notice, so that you can figure out how to rescope your paper or project. But it's hardly my place to tell you when to feel overwhelmed: just make sure you speak up when you are.

Take breaks (every day, and every year)

If you enter grad school thinking you will have a 9-to-5 job, you might have to adjust your expectations. That said, for the same reason you can't skimp on sleep, you shouldn't skimp on walking away from your work: you need breaks.

Before I moved in with my partner, around dinnertime I would walk somewhere close to school for food, eat it on the way back to lab, and keep trying to bang my head against the pre-dinner problem, usually with less-than-stellar results. With some frustration, I'd head home tired and go to sleep. I'd wake up cranky with memories from the night before, and make just enough headway thanks to the break that sleep gave me. It wasn't a very happy way to be.

One of the most convenient side effects of moving in with a partner who finished work around dinnertime was that it forced me to stop working, head home, eat and talk while not working, and then decide what to do after dinner was over. Sometimes, if inspired, I'd work on something I thought of on the walk home. Other times, my mind was trashed and I did something other than research before going to bed, giving me a better shot at the next day.

There's another kind of break you should take, and this one is half-funded by the type of work you do. Conferences tend to be in places you haven't been, with travel paid for by a funding agency. Take the conference location as a randomizer for your vacation each year, and buy your return plane ticket for two weeks after the conference ends. (Don't be ridiculous, and make sure you pay for all non-conference expenditures from your pocket). Two weeks of vacation time might seem insane, but it's important to spend some time each year to avoid email, take a break from thinking about the problems you see day-in-day out, and sidestep the stress of having to produce results weekly. I find that on these vacations, I end up getting bored and thinking about problems related to the ones I'm working on, albeit without the stress, and come back not only re-energized to work on my current problem, but with a few more ideas for future ones.

Also: remember weekends. Unless you have a major deadline coming up, weekends are for resting and having a different set of experiences, not for toiling away at the same problem you do during the week. It might be the case that you feel like working. Don't stop yourself from doing this, but try to do something different than what you did on Friday. If you were programming on Friday, try to write. If you were thinking through the early stages of a project, try to finish up a side project. If a weekend comes and you don't have to work but you choose to, try to work on something new or different.

Unlike our friends in the life sciences, where a mouse's reproduction schedule or the schedule of a several-billion dollar experiment dictates when you work, most research in Computer Science is dictated by when you choose to use your brain. Choose wisely.

Sleep

You are a knowledge worker. Unlike machines in a factory, it's not always the case that working for an extra hour results in an extra hour of output. This is particularly true at 3 am in your 21st hour of being awake.

Early in grad school, I would go to sleep at 3 am and wake up at 8 am every day, thinking this would maximize my productive time. Instead, I hit refresh on my RSS reader and email a lot, having little energy to work through original thought.

I hit several forms of depression working this way, and decided after completing my Master's thesis that if I couldn't be successful in grad school AND treat my body with respect, I should give up on grad school and not my body. In the years following this decision, I slept around seven hours a night, exercised more, and took breaks from work more often. I published more papers and contributed more open source code per year than under my work-till-your-drop regime. You should sleep more and treat yourself better for your own health. But if you don't do it for your health, do it for your research!

What about deadlines? A grad student's calendar is filled with month-long streaks with no pending deadlines. It's also dotted with last-minute heroic efforts to spruce up a paper for the three to four conference deadlines that matter to your field every year. In essence, without serious self-control and planning, there will be periods of your year that are more hectic than others. You should still sleep during those periods. If there is a week left on your deadline, pulling an all-nighter virtually guarantees you will not do good work on the paper/experiments the next day, so you may as well sleep and wake up early.

There's one exception that has worked reasonably for me: if your conference deadline is at 8 am, it might be OK to stay up a bit later the night of the deadline since you can submit, go to sleep, and not have anything major to do when you wake up. If you pull an all-nighter anything earlier than the night of the deadline, you're essentially trading well-rested work time for bleary-eyed hacking/editing. Don't do that.

Peer review

This document is more than crappy first draft, but it's also not the whole story. Please fork it on github and write your own. Some early feedback from friends provided some more voices on the topic:

- Ted Benson identified an important link between "Make something cool" and "Start the conference circuit early." Not only do you have to find something cool, but so do the people who read your papers and see your talks. It helps to meet those people and hear their thoughts sooner than later.
- Michael Bernstein had a few points on the limitations and implications of some of my advice, and a collection of awesome people's thoughts on the topic:
    - He pointed out a side-effect of writing early: "Another outcome of the 'crappy first draft' philosophy: turns out, this kind of write-before-you-code approach is pretty much like what you do later when you're faculty and need to write a grant."
    - He drew the line between after-the-fact tests for your accomplishments and advice that actually helps you finish: "'Make cool things' is a litmus test, but not a process. That's the tough bit. I like David Kelley's quote, 'Enlightened trial and error outperforms the planning of flawless intellect.'"
    - He raised the point that we should focus on our strengths: "I really liked Martin Rinard's advice to try to find something that takes advantage of your strengths, and then teach yourself to be just-good-enough in all other areas. You don't have to be brilliant at everything."
    - Finally, he suggested that grad school is a place to learn what makes for good research: "Eytan Adar said, and I agree with: a big part of grad school is developing a 'taste' for good problems. It's hard to describe how one does that."
- Neha Narula said it might be nice to hear some more prescriptive advice on picking topics and doing the research: "Personally, I'd love to hear more about how you decided what was worth working on, and how you dealt with prototyping and building new things. Was it easy? Do you love programming? How bad was debugging? How did you make it through?"
- Eugene Wu recommended a different framing of this text, which would be amazing if someone did it: a

year-by-year summary of what to expect/do as you advance through grad school. He also came up with a decent number of rap lyrics with which I could prefix each section, but I didn't have the heart.

- Jacob Eisenstein made several great points about what matters, and about how I might have gotten lucky by being at a funding-rich school like MIT (direct quotes below):
  - "There's another guide to grad school from two of my colleagues."
  - "This is implicit in what you wrote, but I would add: come to lab every day, and make friends there, even if you think you are too cool for school. This will keep grad school from being miserable, and will make you a better researcher too."
  - "Authorship and authorship order matter a lot less than you (the typical grad student) think they do. If you do the work, make sure you get some credit. If you think someone else is getting too much credit for your work, don't be a doormat, but remember that this isn't a zero-sum game."
  - "Learn to communicate with your advisor. If something doesn't work, learn to explain why it doesn't work. If something does work, explain how you are certain that it works. If there's something you want to try, explain why it's a cool idea. This often involves additional "pilot" experiments. Try to think of them by yourself, rather than waiting for your advisor to tell you. The amount of freedom that you get as a researcher will be directly proportional to your advisor's confidence that you share an understanding about how good research works."
  - "If your advisor asks you to do something, either (1) do it, or (2) explain very clearly why you don't want to do it (or can't do it). This may seem obvious, but you would be absolutely amazed how many students don't follow this advice."
  - Jacob also added that just making something cool might be more difficult at schools that are less funding-rich than MIT. Some research is tied to a small number of specific grants, and being a grad student in such a lab suggests that you're going to work on a more well-defined piece of a larger puzzle.
- Colin Scott has some great advice for folks who want to make things that matter to practitioners: "[D]o internships. Getting your hands on problems faced by practitioners is deeply valuable."
- Jaime Teevan agrees with Colin. By interning, you can meet new people, see different work environments, try new projects, and work on new papers!
- Nick Crawford has a few more recommendations: "I would add 'managing up', listening to people, being excited about science, and socializing outside of Uni as important 'skills.'"

Thank you

Thanks for reading this. I'd love to speak with you about it. Remember that you're wonderful. Remember to make something cool. Remember to not do it alone.

I'm grateful to Ted Benson, Michael Bernstein, Meredith Blumenstock, Neha Narula, and Eugene Wu for reading through versions of this document and making it better. Grad school was what it was because of my amazing advisors: Sam Madden, David Karger, and Rob Miller.

Finally, a big thank you to the fine folks who made bootstrap, whose code I used to make this guide, and whose design I ripped off and never looked back.