Faster Than You Think

ESSAYS ON THINKING BETTER AND BUILDING FASTER

Daniel Lemire

Contents

Preface 1

How to Think, Learn, and Produce 3

We see something that works, and then we understand it 7

Good students find questions, not answers 11

Research should not stop with the research paper 13

How long should you work on a problem? 17

To improve your intellectual productivity 21

Are you really running out of time? 25

Why speed matters 27

Negative incentives in academic research 29

The ivory tower's drift 33

Thinking clearly 35

Graduate degrees are overrated 39

How I debate 41

Is research sick? 43

How to learn efficiently 45

Why measuring productivity is hard 49

Other useless school trivia: the quadratic formula 51

My kindergarten story 55

Don't study Latin if you want to become a better programmer 57

Why I never give straight answers 59
Elegance as a luxury 63
Does academic research cause economic growth? 67
Science is the belief in the ignorance of experts 73
Enough with the bogus medical studies! 81

The Future Is Malleable 87
The case for techno-optimism 91
Technology sets the bar higher 95
The courage to face what we do not understand 99
Beyond papers 103
Will emerging artificial intelligence boost research productivity? 105
Embodied cognition and agentic AI 109
Antifragile programming and why AI won't steal your job 115
AI: igniting the spark to end stagnation 117
Let us bury the linear model of innovation 123
Models and science 127
As smart as a PhD 129
Sovereign 131
Foolish enough to leave important tasks to a mere human brain? 135

The Craft of Software 141
Does software performance still matter? 145
Hotspot performance engineering fails 153
Metcalfe's law against Brooks' law 157
Don't be so eager to rewrite your code 159
Fast software is a discipline, not a purpose 161
Don't argue: build 163
Programmer time and the pitfalls of false positives 165
Afterword 169

About the Author 171

Preface

These forty-three essays and two interludes span nearly two decades of thinking about research, productivity, technology, and the craft of software. They were written at different moments, for different audiences, and on different subjects. What unites them is a single temperament: respect evidence, distrust slogans, and keep building.

The book has two main parts and a short closing section. Part I examines how to think, learn, and produce serious work over a long career. Part II argues that the future is more malleable than pessimism allows—from scientific institutions to artificial intelligence and what comes next. The closing section turns that stance toward software: performance engineering, rewriting versus refining, and the habits that separate durable systems from fragile ones.

I have edited lightly. Dates appear at the end of each chapter so you can see when a piece was written. Occasional references to products, companies, or tools may feel dated; I have left them when they clarify the argument. The goal is not a time capsule, but a coherent book that reads well on its own.

If you finish with two convictions, I will be satisfied: that clear thinking compounds over a career, and that we routinely underestimate how much room remains to improve the world.

How to Think, Learn, and Produce

These twenty-three essays—and a short interlude—examine intellectual productivity: how to choose problems, how to learn hard material, and how to stay effective over a long career. No specialized background is required—only curiosity about how good work is actually done.

Research and thinking. The opening chapters include *We see something that works, and then we understand it* (2025) and *Good students find questions, not answers* (2020)—on how progress often begins with a working example rather than a theory, and on the art of finding the right problem. Later essays extend that thread through research practice, academic incentives, clear thinking, and work that matters beyond the paper.

Productivity and learning. *Are you really running out of time?* (2008) argues that creative people run out of energy before they run out of hours, and *Why speed matters* (2025) presses the case for moving as close as possible to the speed of your own thought. *How to learn efficiently* (2014) distills active struggle, interleaved practice, and real projects over passive consumption. Companion essays round out the section on intellectual method, graduate training, debate, and honest measurement.

We see something that works, and then we understand it

"We see something that works, and then we understand it." (Thomas Dullien)

It is a deeper insight than it seems.

Young people spend years in school learning the reverse: understanding happens before progress. That is the linear theory of innovation.

So Isaac Newton comes up with his three laws of mechanics, and we get a clockmaking boom. Of course, that's not what happened: we get the pendulum clock in 1656, then Hooke (1660) and Newton (1665–1666) get to think about forces, speed, motion, and latent energy.

The linear model of innovation makes as much sense as the waterfall model in software engineering. In the waterfall model, you are taught that you first need to design every detail of your software application (e.g., using a language like UML) before you implement it. To this day, half of the information technology staff members at my school are made up of "analysts" whose main job is supposedly to create such designs based on requirements and supervise execution.

Both the linear theory and the waterfall model are forms of thinkism, a term I learned from Kevin Kelly. Thinkism sets aside practice and experience. It is the belief that given a problem, you should just think long and hard about it, and if you spend enough time thinking, you will solve it.

Thinkism works well in school. The teacher gives you all the concepts, then gives you a problem that, by a wonderful coincidence, can

be solved just by thinking with the tools the same teacher just gave you.

As a teacher, I can tell you that students get really angry if you put a question on an exam that requires a concept not explicitly covered in class. Of course, if you work as an engineer and you're stuck on a problem and you tell your boss it cannot be solved with the ideas you learned in college… you're going to look like a fool.

If you're still in school, here's a fact: you will learn as much or more every year of your professional life than you learned during an entire university degree—assuming you have a real engineering job.

Thinkism also works well in other limited domains beyond school. It works well in bureaucratic settings where all the rules are known and you're expected to apply them without question. There are many jobs where you first learn and then apply. And if you ever encounter new conditions where your training doesn't directly apply, you're supposed to report back to your superiors, who will then tell you what to do.

But if you work in research and development, you always begin with incomplete understanding. And most of the time, even if you could read everything ever written about your problem, you still wouldn't understand enough to solve it. The way you make discoveries is often to either try something that seems sensible, or to observe something that happens to work—maybe your colleague has a practical technique that just works—and then you start thinking about it, formalizing it, putting it into words… and it becomes a discovery.

And the reason it often works this way is that "nobody knows anything." The world is so complex that even the smartest individual knows only a fraction of what there is to know, and much of what they think they know is slightly wrong—and they don't know which part is wrong.

So why should you care about how progress happens? You should care because… 1. It gives you a recipe for breakthroughs: spend more time observing and trying new things… and less time thinking abstractly. 2. Stop expecting an AI to cure all diseases or solve all problems just because it can read all the scholarship and "think" for a

very long time. No matter how much an AI “knows,” it is always too little.

References

Godin, B. (2017). *Models of innovation: The history of an idea*. MIT Press.

Originally published · December 4, 2025

Good students find questions, not answers

It is often believed that learning is a simple matter of collecting answers and replies. I suspect that "learn mechanistically how to answer the questions" is a great way for weak students to pass courses, and for smart students to ace courses. However, I believe that if you really want to learn the material, you have to learn to ask good questions and stop focusing on answers.

If you ask experienced professors, they will probably admit that some students who do very well on tests (A+ all the way) make uninspiring graduate students, while less stellar undergraduate students can end up being superb graduate students.

Part of the difference is that in graduate school, you have to ask questions first. A researcher's job is to come up with new questions, not necessarily new answers. That is because once you have a good question, the answer can often be derived mechanistically.

It is somewhat counterintuitive if you have been trained in conventional schools. For example, you see mathematics as a set of "obvious questions" followed by "non-obvious answers". And you think that, surely, the hard part was coming up with the answers.

This leads you to think that the way forward is to work really hard on really complicated answers to simple questions. But where will the questions come from?

What is a good question?

1. A question should be at the frontier of the domain of knowledge as you understand it. To really master the material, you have to go beyond the surface and understand its limits.

2. It should trigger interest on the part of the recipient. Your question should be interesting, not just to you, but to the person receiving the question. You may wonder why it should matter: it does because it puts a psychological bar, ruling out tempting but shallow questions.

A good question is usually prepared, not improvised. Most people need a few minutes to come up with a good question. But there are some quick recipes that often deliver good questions…

- Questions about definitions… It is common that a speaker will use terms without defining them clearly. If you are unsure about how you would define a term, it might make for a great question.
- "How do you know" questions are often great. If the expert tells you that programming in a given manner reduces memory usage, you may ask "How would I know?"

The real reason for asking good questions is that the process itself will force you to learn. That is, if you are interested in learning, in really learning, you will not focus on coming up with answers; rather, you will focus on coming up with questions.

You can see it play out if you pay attention. Go to an event where "expert learners" are trying to up their game. For example, a good workshop with people at the top of their game. If the event is well managed, you will observe that people are asking each other good questions.

Experienced professors do it effortlessly. The best engineers are great at it.

As an experiment, next time you encounter an expert in a field that you do not yet master, focus on trying to come up with good, non-obvious questions. I bet that you will learn a lot.

Originally published · September 7, 2020

Research should not stop with the research paper

The practice of academic research is based on the production of formal documents that undergo formal review by peers. We routinely evaluate academics for jobs and promotions based on their publication output. When asked about their contribution to science, many academics are happy to point at their papers. In some cases, they will also point at the grants that they have secured.

Back when I worked at NRC, a friend of mine, Martin Brooks, gave a controversial talk entitled "user-based research" (Brooks, ca. 2005). It has been nearly 20 years, and I still remember this talk. His ideas were so upsetting that people left while he was talking. I stayed and listened. It took me years to own the ideas he expressed that day.

Martin complained that researchers mostly "threw research over the wall" expecting that other people (maybe industry) would pick up the research from the other side of the wall. While it certainly can happen that others will read your paper and apply it, you should not count on it. Nobody has to read and understand your work.

When I was a bit younger, a senior professor pointed out that some idea we were discussing had been described and tested by his team in some prestigious venue decades ago *but that nobody ever did anything with it.* Without thinking, I replied: *it was your job to do something with it, don't blame others*. The room fell silent and there was a long pause.

I am not saying that if you find a way to cure cancer in mice, it is your job to get the therapy cleared for use with human beings and to open a hospital where the therapy is delivered. A single individual can only do so much.

What I am saying, however, is that publishing a research paper is not the goal of research. It is not the final output. It is only one element in a chain. And it is not even a requirement. You need to communicate your idea, but the peer-reviewed journal article is just one such means.

Your actual goal is "transfer". That is, someone, somewhere, must put your ideas into practice beyond the publication of your paper. It does not mean "industrial applications" though it can be that. If your idea is worthwhile and you let it end with a research paper, you have failed.

And it does not merely mean collecting "citations" or other bibliometrics. People routinely cite papers without reading them. Few citations are influential (Priem & Costello, 2015).

But the academic incentives almost conspire to prevent impactful research. There is one specific criterion that academics like to apply that is destructive: novelty. For some piece of academic work to be judged worthwhile, it must be novel. I will say it again and again: originality is overrated.

Of course, people entering a new field tend to "rediscover known facts". You have to push them back and tell them to go study some more. But there is a difference between naïveté and lack of originality. You have to be aware of the history of your field, that is what scholarship is all about. But you also have to stick with ideas for the long run, until the fruits appear.

Instead of rewarding novelty, we should reward scholarship: we should ask people to show that they have studied and documented the past. We should never penalize someone who works on pushing a known idea by refining it, communicating it, and validating it.

This idea that some venerable professor had 20 years ago that never went anywhere? Well, it might be entirely worthwhile to revisit it and make it go somewhere, even if it is not novel at all.

References

Boote, D. N., & Beile, P. (2005). Scholars before researchers: On the centrality of the dissertation literature review in research preparation. *Educational Researcher*, 34(6), 3–15. https://doi.org/10.3102/0013189X034006003

Priem, J., & Costello, K. L. (2015). Few citations are influential. *arXiv*. https://doi.org/10.48550/arXiv.1501.06587

Brooks, M. (ca. 2005). User-based research [Unpublished presentation]. National Research Council Canada.

Originally published · February 7, 2020

How long should you work on a problem?

Lev Reyzin (2021) says that working too long on a problem might be unproductive:

> I, personally, have diminishing (or negative?) returns to my creative work as I explicitly work on a problem past some amount of time. I often have insights coming to me out of nowhere while I'm relaxing or enjoying hobbies on nights or weekends.

Whenever one considers innovative endeavors and their productivity, one must consider that innovation is fundamentally wasteful. The problem with innovation is not about how to get as much of it for as little of a cost as possible. It is to get innovation at all. By optimizing your production function, you risk losing all of it. *I sooner blame someone for his publication list being too long than being too short*, said Dijkstra (1982).

My view is that we tend to underestimate "intellectual latency". There is a delay between the time you approach a new idea and the time you have fully considered it.

Thus our brains are not unlike computers. Your processor might be able to run at 4 GHz and be able to retire 4 instructions per cycle… but very rarely are you able to reach such a throughput while working on a single task. The productive intellectuals that I know tend to work on a few ideas at once. Maybe they are writing a book while building a piece of software and writing an essay.

So you should not focus on one unique task in the hope of finishing it faster. You may complete it slightly faster if you omit everything else but the sum total of your productivity might be much lower.

There is also a social component to human cognition. If you hold on to a problem for very long, working tirelessly on it, you may well deprive yourself of the input of others. You should go work and then quickly invite others to improve on your work. No matter how smart you think you are, you cannot come close to the superior ingenuity of the open world.

Energy and sanity are essential ingredients of sustained intellectual productivity. Hammering at a single problem for a long time is both maddening and energy limiting. Our brains are wired to like learning about new ideas. Your brain wants to be free to explore.

And finally, the most important reason to limit the amount of work you invest in a single task is that it is a poor strategy even if you can do it with all the energy and intelligence in the world. Sadly, most of what you do is utterly useless. You are like an investor in a stock market where almost all stocks are losers. Putting all your money on one stock would ensure your ruin. You cannot know, at any given time, what will prove useful. Maybe going outside and playing with your son sounds like a waste of your potential right now, but it might be the one step that puts your life on a trajectory of greatness. You want to live your life diversely, touching many people, trying many things. Learn to cook. Make cocktails. Dance. Go to the theater. Play video games. Write assembly code. Craft a novel.

Many years ago I started to blog. I also started publishing my software as open source in a manner that could be useful to others. I started posting my research papers as PDFs that anyone could download. None of these decisions seemed wise at first. They took time away from "important problems". I was ridiculed at one point or another for all of them. Yet these three decisions ended up being extremely beneficial to me.

References

Reyzin, L. [@lreyzin]. (2021, June 14). I, personally, have diminishing (or negative?) returns to my creative work as I explicitly work on a problem

past some amount of time [Post]. *X*. https://twitter.com/lreyzin/status/1404496460809441280

Dijkstra, E. W. (1982). *Selected writings on computing: A personal perspective*. Springer-Verlag.

Originally published · June 14, 2021

To improve your intellectual productivity

We would all like to be smarter, to produce better software, better research papers or better art. It is not difficult to see that, by just about any metric, productivity amongst human beings follows a power law. For example, 80% of all research papers are written by 20% of the researchers. (I made this one up, but the actual numbers are similar: see Lotka's law (Lotka, 1926).)

However, it is a difficult topic to study because the very idea of measuring intellectual productivity is, at best, controversial.

Yet we can't help but ask: what does determine your intellectual productivity? Here are some tentative answers:

- **Luck**. A lot of intellectual work is based on trial and error. A couple of early breakthroughs, due to pure chance, can motivate you and inspire you to work for years. Repeated failures can discourage you and push you to abandon leading edge work. Luck alone can explain the power law in the distribution of intellectual productivity.
- **Persistence**. Some people are pig headed. I sure am. They will keep working even when things are not working. This is closely related to the willingness to take chances. Most great intellectuals dive in and take risks.
- **People**. Some people drain you without helping you produce. Other people energize you and force you to improve your game. Even though I am an introvert and sometimes an insular person, I have not found a better way to boost my productivity than finding exactly the right people at the right time. Sometimes I will have worked for months on a problem just to have several breakthroughs

because I started working with someone smart. The quality of my work can also be strongly affected by the people I work with: people with higher standards than mine tend to push me up, people with lower standards tend to make me more lenient. But it is not just a matter of meeting smart people. I do not think I would get a lot out of meetings with Einstein. I am not interested in Physics right now and I doubt Einstein would share my interests. You need to meet people who have truly compatible goals and interests. Some people have an innate ability to connect with others, to find the right collaborators. Others spend a lot of time meeting people and are therefore more likely to find good matches. Though I have been extremely lucky in this respect for the last few years, collaboration is not and will never be my strong point.

- **Method**. Some people pay attention to what works and to what does not work. They experiment. They read about how great people worked. They learn meta-strategies like divide-and-conquer. They constantly tune their approach to intellectual productivity. Some people figure out that working seriously for a few hours every day is better than doing pseudo-work all day. I find that a common source of low productivity is excessive pondering: you cannot be productive if you fail to deliver in time. Other people just stumble on the right approach early on. For example, ever since I was a teenager, I have collected ideas in a notebook. This means that I am never out of crazy (and usually bad) ideas.
- **Focus**. Intellectual work is boring in a way that manual labor is not. Intellectual work can become frankly alienating. Some people seem to never get bored. They can do narrow mathematics for days and days without ever getting bored. They are satisfied by the work itself. I have weak boredom resilience myself: I constantly question the relevance of my work. While it sounds good, it can actually be quite bad as you can be far more productive if you just keep your head down and push forward. I also see a lot of people who willingly engage in low productivity activities. Then they wonder why they aren't being productive! A lot of *work* is

actually *busy work*: you feel as if you were working, but you aren't producing lasting value.

- **Fire**. You have to care to have some intellectual productivity. You must feel bad when nothing was achieved. And I mean really bad. Some people are only driven by fear of losing their financial support. Others cannot stand not producing anything intellectually. They do not wait for permission nor do they plan to retire. These people, those who feel an urge to produce, will naturally be much more productive.

References

Lotka, A. J. (1926). The frequency distribution of scientific productivity. *Journal of the Washington Academy of Sciences*, 16(12), 317–323.

Originally published · August 24, 2012

Are you really running out of time?

A common feeling among creative workers is the **lack of time**. Yet, most people will run out of energy before they run out of time. A single task that takes you 5 minutes (asking a Business Development Officer for Intellectual Property rights) can drain you out for a week. Another task, like lecturing for 3 hours, can energize you for the rest of the week. Highly productive people do not have more time, but they may have more energy, more method and better feedback on their progress.

I believe that three problems lead us to conclude we lack time:

- **You are spending too much time on boring tasks.** To be productive, you need to work on projects you love. For this reason, creative people should pick their projects.
- **You fail to manage your projects**. Without help, you can only keep track of about 7 projects or tasks at any one time. If you want to do more, a method is needed. Myself, I use Getting Things Done (Allen, 2001). But some method is needed to scale up to a large number of projects. Without method, you will drift to unessential tasks and then blame the lack of time to explain why important tasks went unattended.
- **You do not measure your progress**. You need to get feedback about the quality and quantity of your work. Myself, I put my work under version control and get daily emails of what files changed. It is a crude but effective measure of my work. Also, tracking your project carefully, at the task level, helps. Finally, having coworkers who react to your work is a blessing. Without

a measure of your progress, you may realize too late that your projects did not progress and then blame the lack of time.

References

Allen, D. (2001). *Getting things done: The art of stress-free productivity*. Penguin.

Originally published · November 30, 2008

Why speed matters

The one constant that I have observed in my professional life is that people underestimate the need to move fast.

Of course, doing good work takes time. I once spent six months writing a URL parser. But the fact that it took so long is not a feature, it is not a positive, it is a negative.

If everything is slow-moving around you, it is likely not going to be good. To fully make use of your brain, you need to move as close as possible to the speed of your thought.

If I give you two PhD students, one who completed their thesis in two years and one who took eight years… the two-year thesis is usually the healthier signal—not always, but often enough that speed is information.

Moving fast does not mean that you complete your projects quickly. Projects have many parts, and getting everything right may take a long time.

Nevertheless, you should move as fast as you can.

For multiple reasons:

1. A common mistake is to spend a lot of time—too much time—on a component of your project that does not matter. I once spent a lot of time building a podcast-like version of a course… only to find out later that students had no interest in the podcast format.
2. You learn by making mistakes. The faster you make mistakes, the faster you learn.
3. Your work degrades, becomes less relevant with time. And if you work slowly, you will be more likely to stick with your slightly obsolete work. You know that professor who spent seven years preparing lecture notes twenty years ago? He is not going to throw them away and start again, as that would be a new seven-year

project. So he will keep teaching using aging lecture notes until he retires and someone finally updates the course.

What if you are doing open-heart surgery? Don't you want someone who spends days preparing and who works slowly? No. You almost surely want the surgeon who does many, many open-heart surgeries. They are very likely to be the best one.

Now stop being so slow. Move!

Originally published · December 5, 2025

Negative incentives in academic research

In the first half of the 20th century, there were relatively few scientists, and these scientists were generally not lavishly funded. Yet it has been convincingly argued that these scientists were massively more productive. We face a major replication crisis where important results in fields such as psychology and medicine cannot be reproduced independently. Academic researchers routinely fail to even fill out the results from clinical trials. We have entered a 'dark age' where we are mostly stagnant. It is not that there is no progress per se, but progress is slow, uncommon and expensive.

Why might that be? I believe that it has to do with important 'negative incentives' that we have introduced. In effect, we have made scientists less productive. We probably did so through several means, but two effects are probably important: the widespread introduction of research competitions and the addition of extrinsic motivations.

1. Prior to 1960, there was hardly any formal research funding competitions. Today, by some estimates, it takes about 40 working days to prepare a single new grant application with about 30 working days for a resubmission, and the success rate is often low, which means that for a single successful research grant, hundreds of days might have been spent, purely on the acquisition of funding. This effect is known in economics as rent dissipation. Suppose that I offer to give you $100 to support your research if you enter a competition. How much time are you willing to spend? Maybe you are willing to spend the equivalent of $50, if the success rate is 50%. The net result is that two researchers may each waste $50 in time so that one of them acquires $100 of support. There may be

no net gain! Furthermore, if grant applications are valued enough (e.g., needed to get promotion), scientists may be willing to spend even more time than is rational to do so, and the introduction of a new grant competition may in fact reduce the overall research output. You should not underestimate the effect that constant administrative and grant writing might have on a researcher: many graduate students will tell you of their disappointment when encountering high status scientists who cannot seem to do actual research anymore. It can cause a vicious form of accelerated aging. If Albert Einstein had been stuck writing grant applications and reporting on the results from his team, history might have taken a different turn.

2. We have massively increased the number and importance of 'extrinsic motivations' in science. Broadly speaking, we can distinguish between two types of motivations… intrinsic and extrinsic motivations. Winning prizes or securing prestigious positions are extrinsic motivations. Solving an annoying problem or pursuing a personal quest are intrinsic motivations. We repeatedly find that intrinsic motivations are positively correlated with long-term productivity whereas extrinsic motivations are negatively correlated with long-term productivity. In fact, extrinsic motivations even cancel out intrinsic motivations. Extrinsically motivated individuals will focus on superficial gains, as opposed to genuine advances. Of course, the addition of extrinsic motivations may also create a selection effect: the field tends to recruit people who seek prestige for its own sake, as opposed to having a genuine interest in scientific pursuits. Thus creating prestigious prizes, prestigious positions, and prestigious conferences, may end up being detrimental to scientific productivity.

Many people object that the easiest explanation for our stagnation has to do with the fact that most of the easy findings have been covered. However, at all times in history, there were people making this point. In 1894, Michelson said:

> While it is never safe to affirm that the future of Physical Science has no marvels in store even more astonishing than those of the past, it seems probable that most of the grand underlying principles have been firmly established and that further advances are to be sought chiefly in the rigorous application of these principles to all the phenomena which come under our notice.

Silverstein in "The End is Near!": The Phenomenon of the Declaration of Closure in a Discipline, documents carefully how, historically, many people predicted (wrongly) that their discipline was at an end.

References

Bloom, N., Jones, C. I., Van Reenen, J., & Webb, M. (2018, November 1). Are ideas getting harder to find? *The Atlantic*. https://www.theatlantic.com/science/archive/2018/11/diminishing-returns-science/575665/

Herbert, R., et al. (2013). The time needed to prepare grant applications: An international survey. *BMJ Open*, 3(5), e002800. https://doi.org/10.1136/bmjopen-2013-002800

Horodnic, I. A., & Zaiţh, A. (2015). Motivation and research productivity in a university system undergoing transition. *Research Evaluation*, 24(3), 282–292. https://doi.org/10.1093/reseval/rvv010

Wrzesniewski, A., Schwartz, B., Cong, X., Kane, M., Omar, A., & Kolditz, T. (2014). Multiple types of motives don't multiply the motivation of West Point cadets. *Proceedings of the National Academy of Sciences*, 111(30), 10990–10995. https://doi.org/10.1073/pnas.1405298111

Silverstein, A. (1997). "The End is Near!": The phenomenon of the declaration of closure in a discipline. *Current Anthropology*, 38(3), 370–372. https://doi.org/10.1086/204623

Originally published · July 21, 2022

The ivory tower's drift

Almost all of academic science has moved away from actual (empirical) science. It is higher status to work on theories and models. I believe that it is closely related to well-documented scientific stagnation as theory is often ultimately sterile.

This tendency is quite natural in academia if there is no outside pressure... And is the main reason why academia should be ruthlessly judged by practitioners and users. As soon as academia can isolate itself in a bubble, it is bound to degrade.

It is worth trying to understand some of the factors driving this degradation... Theoretical work can sometimes be seen as more complex. This complexity can be mistakenly equated with higher intelligence or prestige. Empirical work, while also complex, often deals with tangible, observable data, which might seem more straightforward to the uninitiated.

Empirical work is more likely to lead to nuanced or inconclusive results while theory is often seemingly more direct and definitive. Theoretical research often requires fewer resources than large-scale empirical studies which might need extensive funding for equipment, data collection, and personnel. Thus you get to do more research with less using models and theory.

Theoretical work is often seen as requiring a high level of creativity to devise new frameworks or models. While empirical work also requires creativity in design, execution, and interpretation, the creativity in data collection or experimental design might be less recognized or appreciated.

The educational system often glorifies theoretical knowledge over practical skills until one reaches higher education or specialized training. E.g., we eagerly make calculus compulsory even if it has modest

relevance in most practical fields. This educational bias can carry over into professional work.

We still have laboratories. We still collect data, but it is too often in service of theory rather than a primary activity.

Society must demand actual results. We must reject work that is said 'to improve our understanding' or 'to lay a foundation for further work'. We must demand cheaper rockets, cures for cancer, software that is efficient. As long as academic researchers are left to their own devices, they will continue to fill the minds of the young with unnecessary models. They must be held accountable.

Originally published · January 15, 2025

Thinking clearly

If you have ever met me in person, you know that when you share an idea with me, I simplify it to its core and reflect it back to you, focusing on its essential parts. I dissect each statement for precision. "What do you mean by this word?"

I have two decades of experience working with academics who overcomplicate everything. Humans are easily confused. A project proposal with ten moving parts and five objectives is overwhelming. Most people cannot think it through critically, which can lead to disaster.

By instinct, I simplify problems as my first step, reducing them to their "minimum viable product," as they say in Silicon Valley.

Some people avoid simplicity to sound smarter. They won't admit it, maybe not even to themselves, but that's what they are thinking: "Oh no! I'm not doing this simple thing; my work is much more sophisticated."

That's a terrible idea. Even simple projects become challenging if you are ambitious. There is no need to complexify them. For example, seven years ago, we aimed to create a JSON parser faster than anything on the market—a simple idea. A senior colleague in computer science saw me at a campus coffee shop while I was working. He asked what I was busy doing… When I told him, "We're writing a fast JSON parser," he laughed. He would later admit that he thought I was joking: how could I work on something so mundane. I wasn't kidding. The result was simdjson, a JSON parser four times faster than anything else at the time. By keeping our project conceptually simple, we made success easier.

Complexity is a burden, not a badge of pride.

Clear thinking demands precision.

People use emotionally charged words like “safe.” “My car is safe.” “My software is safe.” What does that mean? Define it precisely.

A graduate student of mine recently proposed reducing the cognitive load of agentic AI on developers. “Cognitive load” sounds great—thousands of papers discuss it—but what does it mean? How do you measure it?

Take “AI” as another example. Nobody knows what AI is. Is a Google search AI? Is image search AI? Is an expert system AI? Or do you mean large language models? Clarify what you mean.

Too often, we accidentally hide behind overly abstract language. Not only does this harm how we think, but it also harms how we are perceived. People who avoid jargon are viewed as more honest, trustworthy and benevolent (Fick et al., 2025).

Your motivation should be clear from the start. Here is an example. I am often asked: “Should I go to graduate school?” What’s the motivation? Often, it’s to get a great job. Years ago, I had this conversation with a research assistant:

- Should I get a Master’s degree? Everyone with good grades does it.
- What is your objective?
- I want a good, well-paying job.

Given his exceptional technical skills, I told him he could already land that job soon. He did, and now he’s in a leadership role at one of Montreal’s top companies. He is outearning me by a wide margin, no doubt.

People often lose sight of their motivation and follow trends. Reconnect with your motivation and adjust your means accordingly.

To sum up, my advice for clear thinking:

- Simplify projects to their essentials early and often.
- Use precise language. Avoid jargon.
- Focus on your motivation first, then choose the appropriate means.

References

Fick, S. J., et al. (2025). The language of trust: How jargon affects perceptions of honesty, trustworthiness, and benevolence. *Journal of Experimental Psychology: General*. https://doi.org/10.1037/xge0001604

Originally published · October 26, 2025

Interlude

Cognitive biases

One-sided bet: People commonly assume implicitly that their actions may only have good outcomes. For example, increasing the minimum wage in a country may only benefit the poor. Taking a lottery ticket only has the upside of possibly winning a lot of money. Believing in God can only have benefits. And so forth. In truth, most actions are two-sided. They have good and bad effects.

Politician's syllogism: We must do something, this is something so we must do it. "We must fight climate change, we can tax oil, so we must tax oil." If there is a problem, it is important to assess the actions we could take and not believe that because they are actions in response to a real problem, they are intrinsically good.

Confirmation bias: "I believe that there are extraterrestrials, I have collected 1000 reports confirming their presence" (but I am blind to all of the negative evidence). People tend to make up their mind first and then to seek to rationalize their opinion whereas they should do the opposite.

Historical pessimism bias: "Human life was so much better 2 centuries ago!" Yet by almost any measure, human beings have better lives today.

Originally published · December 16, 2020

Graduate degrees are overrated

Though I have many brilliant graduate students, I love working with undergraduate students. And I am not at all sure that you should favor people with graduate degrees, given a choice. Many graduate students tend to favor abstraction over practical skills. They often have an idealized view of the world. Moreover, these students are often consumed by research projects, theses, or dissertations, and the publication of scientific articles, which limits their time for concrete actions. On the other hand, undergraduate students, in my experience, show more enthusiasm for concrete and useful projects, even if they are not prestigious at first glance.

This selection effect is also evident in career choices: those aspiring for direct action are eager to engage in it and are not keen on spending years writing abstract documents. Conversely, those who seek to avoid direct engagement with reality tend to prolong their studies. This effect is noticeable during competitions to recruit professors. It is often surprisingly difficult to find candidates with significant real-world experience.

Advanced degrees, even from prestigious institutions, do not necessarily produce disciplined or objective thinkers. The educational process often rewards imitative thinking over original, exploratory thought, leading to graduates who may excel in absorbing specific knowledge but fail in practical, broader application of concepts. The system guides students towards known outcomes rather than encouraging intellectual risk-taking. Advanced degrees do not require exceptional intelligence but rather the ability to memorize and apply taught material, which might not equip one for the nuanced, critical thinking needed in real-world scenarios.

Irrespective of how smart and educated you are, it is difficult to beat a “can do” attitude. The world is built by people who are eager to get things done as opposed to people who merely want to talk or write. Or, at least, that should be what happens in a healthy society.

Originally published · November 12, 2024

How I debate

Many of us feel that the current intellectual climate is difficult to bear. When I first noticed the phenomenon, people told me that it was because of Donald Trump. He just made it impossible to debate calmly. But now that Trump is gone, the climate is just as bad and, if nothing else, much worse.

Debates are essential in a free society. The alternative to debate is force. Either you convince your neighbor to do as you think they should do, or else you send men with guns to his place.

It is tempting, when you have the upper hand, to use force and aggressive tactics against your opponents. However, this leaves little choice to your opponents: they have to return the favor. And if you both live long enough, chances are that they will.

Civility is a public good. We have to all commit to it.

I do not pretend to be the perfect debater. I make mistakes all the time. However, I try to follow these rules.

1. Do not hope to change people's core stance. This rarely, if ever, happens. That is not why we debate. If someone is in favor of Brexit and you are not, you can argue until you are blue in the face and they won't change their stance. One of the core reasons to debate is to find common ground. People will naturally shy away from arguments that are weak. You can see a debate as a friendly battleground. Once the battle is over, you have probably not taken the other person's moat, but if you did your job, you have enticed them to drop bad arguments. And they have done the same: they have exposed weaknesses in your models. It implies that the debate should bear on useful elements, like arguments and facts. It also implies that debate can and should be productive... even if it never changes anyone's stance.

2. Your goal in a debate is neither to demonstrate that the other person is bad or that you are good. Let people's character out of the debate. This includes your own character. For example, never argue that you are a good person. Reject character assassination, either of yourself or of others. The most popular character assassination tactic is "by association": "your employer once gave money to Trump so you are a racist". "You read this news source so you are part of their cult." You must reject such arguments, whether they are applied to you or to others. Another popular tactic is to question people's motives. Maybe someone works for a big oil company, so that explains why they are in favor of Brexit. Maybe someone is a member of the communist party, and that's why they want to give the government more power. It is true that people's motives impact their opinions, but it has no room in civil debate. You can privately think that a given actor is "sold out", but you should not say it.
3. Shy away from authority-based arguments. Saying that such and such is true because such and such individual says so, is counterproductive because the other side can do the same and the debate will be sterile. You can and should provide references and sources, but for the facts and arguments that they carry, not for their authority.

I believe that a case can be made that without a good intellectual climate, liberalism is bound to fade away. If you want to live in a free society, you have to help enforce good debates. If you are witnessing bad debates, speak up. Remind people of the rules. In fact, if I deviate from these rules, remind me: I will thank you.

Originally published · September 9, 2021

Is research sick?

One of the most important database researchers of all time, Michael Stonebraker, has given a talk recently on the state of database research. I believe that many of his points are of general interest:

- **We have lost our consumers...** Researchers write for other researchers. They are being insular and ultimately irrelevant.
- **We ignore the important problems...** Researchers prefer to focus on the easy problems they know how to solve right away.
- **Irrelevant theory comes to dominate...** Making things work in the real world is harder than publishing yet-another-theory paper.
- **We do not have research taste...** Researchers jump on whatever is fashionable with little regard to whether it makes sense (XML, Semantic Web, MapReduce, Object databases and so forth).

The core message that Stonebraker is trying to pass is that the seminal work he did years ago would no longer be possible today.

The fact that we are publishing many papers is not, by itself, a problem. The irrelevant theory can be ignored. The fact that many people follow trends mindlessly whereas only a few make their own ways, is just human nature.

So what is the real problem?

My take is that there is an underlying supply-and-demand issue. We are training many more PhDs than there are desirable tenure-track jobs. This puts early-career researchers in an over-competitive game. If you can signal your value by getting papers accepted at competitive venues, then that is what you are going to do. If you need to temporarily exaggerate your accomplishments, or find shortcuts, maybe you will do so to secure the much needed job.

You cannot magically make a field less competitive without tackling the supply and demand. Either you reduce the number of PhDs, or

you increase the number of desirable research jobs. I'd vote for the latter. Stonebraker says nothing about this. In fact, though I like him well enough, he is an elitist who cares for little but the top-20 schools in the US. The problem is that even just these top-20 schools produce many more PhDs than they consume. Is it any wonder if all these PhD students desperately play games?

The reason Stonebraker was able to get a tenure-track job in a good school when he completed his PhD without any publication is… Economics 101… there was no shortage of tenure-track jobs back then.

Is it any wonder if the new professors recruited in an over-competitive system are status-obsessed folks?

Originally published · May 14, 2018

How to learn efficiently

I am convinced that much of the gap between the best college students and the worst is explained by study habits. Frankly, most students study poorly. To make matters worse, most teachers are incapable of teaching good study habits.

Learning is proportional to effort

Sitting in a classroom listening to a professor feels like learning… Reading a book on a new topic feels like learning… but because they are overwhelmingly passive activities, they are inefficient. It is even worse than inefficient, it is counterproductive because it gives you the false impression that you know the material. You can sit through lecture after lecture on quantum mechanics. At some point, you will become familiar with the topics and the terminology. Alas, you are fooling yourself which is worse than not learning anything.

Instead, you should always seek to challenge yourself. If some learning activity feels easy, it means that it is too easy. You should be constantly reminded of how little you know. Great lectures make it feel like the material is easy: it probably is not. Test yourself constantly: you will find that you know less than you think.

Some students blame the instructors when they feel confused. They are insistent that a course should be structured in such a way that it is always easy so that they rarely make mistakes. The opposite is true: a good course is one where you always feel that you will barely make it. It might not be a pleasant course, but it is one where you are learning. It is by struggling that we learn.

On this note, Learning Style theory is junk: while it is true that some students have an easier time doing things a certain way, having it easier is not the goal.

There are many ways to challenge yourself and learn more efficiently:

- Seek the most difficult problems, the most difficult questions and try to address them. It is useless to read page after page of textbook material, but it becomes meaningful if you are doing it to solve a hard problem. This is not news to Physics students who have always learned by solving problems. Always work on the toughest problems you can address.
- Reflect on what you have supposedly learned. As an undergraduate student, I found that writing a summary of everything I had learned in a class was one of the best ways to study for an exam. I would just sit down with a blank piece of paper and try to summarize everything as precisely as possible. Ultimately, writing your own textbook would be a very effective way to learn the material. Teaching is a great way to learn because it challenges you.
- Avoid learning from a single source. Studying from a single textbook is counterproductive. Instead, seek multiple sources. Yes, it is confusing to pick up a different textbook where the terminology might be different, but this confusion is good for you.

If sitting docilely in a classroom is inefficient and even counterproductive, then why is it so common a practice? Why indeed!

Interleaved study trumps mass study

When studying, many people do not want to mix topics "so as not to get confused". So if they need to learn to apply one particular idea, they study to the exclusion of everything else. That is called mass (or block) practice.

Course material and textbooks do not help: they are often neatly organized into distinct chapters, distinct sections… each one covering one specific topic.

What researchers have found is that interleaved practice is far superior. In interleaved practice, you intentionally mix up topics. Want to become a better mathematician? Do not spend one month studying combinatorics, one month studying calculus and so on. Instead, work on various mathematical topics, mixing them randomly.

Interleaved practice feels much harder (e.g., "you feel confused"), and it feels discouraging because progress appears to be slow. However, this confusion you feel… that is your brain learning.

Interleaved practice is exactly what a real project forces you to do. This means that real-world experience where you get to solve hard problems is probably a much more efficient learning strategy than college. Given a choice between doing challenging real work, and taking classes, you should always take the challenging work instead.

References

Brown, P. C., Roediger, H. L., & McDaniel, M. A. (2014). *Make it stick: The science of successful learning*. Harvard University Press.

Dunlosky, J., Rawson, K. A., Marsh, E. J., Nathan, M. J., & Willingham, D. T. (2013). Improving students' learning with effective learning techniques. *Psychological Science in the Public Interest*, 14(1), 4–58. https://doi.org/10.1177/1529100612453266

Originally published · December 30, 2014

Why measuring productivity is hard

Studying productivity is challenging. About 15-20 years ago, I was obsessed over my own productivity.

I created a spying agent to monitor my activities and time spent. It would look at which window is opened, and so forth. I would also monitor which web sites were loaded, and so forth.

I discovered that I spent over 40% of my day on email, which was shocking. However, this didn't improve my productivity.

Why is it so hard? The issue is that you're likely not measuring what you think. Productivity is value per unit of time, but defining "value" is problematic.

Long-term, value follows a Pareto distribution (Newman, 2005), where most value is created in unpredictable, short bursts.

For instance, a sudden idea in the shower might lead to an hour of work equivalent in value to the rest of the month's efforts.

Does it mean that you should slack off, wait for the brilliant insight? No. That's the problem with the Pareto distribution in general.

Maybe you are running a company and figure that 20% of your employees do 80% of the work. So you fire the 80% that are less productive. And what happens? Maybe you find out that Joe, who seemed unproductive, was holding your business together and you need to rehire him quickly.

The fundamental issue is that you have limited knowledge. You do not know where the value lies when you are in the middle of it. So by slacking off, you are likely to just greatly diminish the probability that you will have a sudden burst of high productivity.

Long term, you can probably identify the less productive activities. Maybe you have been going to these meetings for two years now, and nothing got done.

But this little side project of yours, that looks like a waste of time, could be (truly) the most important work you could be doing.

So you should be humble and patient. Don't waste time micro-managing yourself or others. You know less than you think.

References

Newman, M. E. J. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46(5), 323–351. https://doi.org/10.1080/00107510500052444

Originally published · July 12, 2025

Other useless school trivia: the quadratic formula

I have two young boys and I have decided to pay attention to what they are learning in school. Beside basic writing and reading skills, mathematics feels like the next most important academic topic. I realize that relatively little has changed in 30 years regarding how mathematics is taught.

As a young kid, I liked mathematics but it became less enjoyable in high school. After a time, I learned to love algebra as it made it possible to solve difficult word problems more easily. To this day, I still do algebra to way I was initially taught, and I often get the right answers.

Later in my schooling, I came across the quadratic formula. That was quite a disappointment. I will not even reproduce it here because it is an ugly hack. Algebra all made sense up to this point, and then we had to memorize a complicated formula. I was not impressed and there were many students to complain about it too. I attended a Catholic school at the time, and I remember that the teacher told us to take it on faith.

Relying on faith to do mathematics did not square well with me. Maybe you can guess that I never did memorize the quadratic formula. Indeed, it is almost trivial and much more satisfying to derive it from first principes. Our teacher was kind enough to rush through the derivation on the blackboard once: it was too fast for any of us to understand, but sufficient for me to figure it out on my own later. How could he derive it at first and then ask us to take it on faith? I guess he thought that having us learn to figure it out on our own would take too long.

I would argue that there is little sense in having people who cannot derive the quadratic formula, memorize it. If they cannot derive it, it is very unlikely that they will grow up to become adults who use algebra in a non-trivial manner. And if they are never going to do algebra, why would they need the quadratic formula?

I think that most people at ease with algebra can derive the quadratic formula quickly. The first thing to note is that there is no reason to find the roots of $ax^2 + bx + c$. You should divide throughout by a.

So, really, all you need to do is to figure out the roots of $x^2 + bx + c$. It is good enough. It is already a more elegant problem.

Then you just need one more trick... it is called "completing the square" and it says that $x^2 + bx = (x + b/2)^2 - b^2/4$. Of course, there is no need to memorize the completion of the square formula...

From this, solving for the quadratic formula is easy... you start from...

$$x^2 + bx = -c$$

and you complete the square...

$$(x + b/2)^2 = -c + b^2/4$$

And that is all!

Admittedly, it can take me longer to solve for the roots of a quadratic polynomial using this derivation than someone who has memorized the formula. I would guess that it takes me an extra 5 seconds. But how often do you have to solve for the roots of such a polynomial without the assistance of a computer?

I will also reiterate my argument against memorizing multiplication tables... Reasoning out that 6 times 8 is 48 is more important than the fact itself. Similarly, figuring out the binomial formula from first principles might help you approach a wider range of problems with confidence...

To be fair to contemporary education, the American Common Core seems to recommend doing exactly as I did as a kid regarding the quadratic formula.

Of course, the quadratic formula was only the first of many disappointments to come. Next we learned trigonometry, and then we had to suffer through analytic geometry…

The pattern would repeat itself endlessly: "You have to learn that that the square of the secant minus the square of the tangent is 1 [replace by your favorite piece of trivia]." Then I would find a way to "route" around the problem. For example, in trigonometry, I eventually figured out that most things could be derived from Euler's formula. That formula itself was intuitive enough if you could remember that the cosine and sine give you the coordinates of a point on the unit circle given an angle. In fact, almost all of trigonometry can be derived from this simple observation.

We still insist on presenting mathematics as a collection of definitions, facts and routines to be memorized. Thankfully, the emphasis is somewhat lesser than in my days… but if you scratch beyond the surface, too little has changed.

Evidently, the purpose of such mathematics in schools is primarily to rank students. It is not very different from the Imperial examination of the Song dynasty. We are teaching to the test because only the test results matter… the test only serves to separate the "good students" from the bad… under a pretence of "merit".

And then, wait for it, we find out that many of our graduates lack skills that would make them employable! It does not matter because employers only look for degrees and diplomas… except when they do not or cannot.

My boys love to play Minecraft. They build all sorts of crazy devices. By my estimation, this is probably a better preparation for the "real world" they will face in 10 years than the math that they are learning in school. Sadly, I am quite serious.

References

Lockhart, P. (2009). *A mathematician's lament: How school cheats us out of our most fascinating and imaginative art form*. Bellevue Literary Press. https://www.amazon.com/Mathematicians-Lament-School-Fascinating-Imaginative/dp/1934137170/

Originally published · March 16, 2015

My kindergarten story

Though I was a straight-A student for most of my high school and college years, I failed kindergarten. I have told this story many times, but never at length in print.

I ended up with a PhD from a top Canadian university and some of my research as an academic has been impactful. I had good grades at the end of high school… I probably was the top student in a large private school.

But, as a kid, I wasn't a good fit for schooling.

I could not tie my shoelaces. You might think that I was particularly lacking in hand-eye coordination, but I don't think that's the case. To this day, there is a good chance that my shoelaces will be undone. I don't know exactly why I am not good with shoelaces. Yet it was one of the tests you had to pass in kindergarten.

The other test you had to pass was to learn your phone number. To this day, I don't know my phone number. I have to look it up, copy and paste it. It is embarrassing because everyone seems to know their phone number. So I failed this test.

You had to learn to count up to 10. I decided that since I was 5, it was fine to only know how to count up to 5. So I failed this test as well.

I probably failed all sorts of other tests as well.

So I was put in a class for students with learning disabilities. It wasn't a punishment, I actually enjoyed it.

My troubles were not over. Like Mandelbrot, I never learned the multiplication tables. Instead I learned tricks to do multiplications in my head. Teachers tried really hard to prevent such an approach and you had to hide your schemes. I also never learned the quadratic

formula, instead I figured out how to complete the square quickly enough.

What I did learn, apparently, is independent thinking. Because I had a difficult relationship with teachers and what they wanted, I think I learned to figure out what I wanted to learn on my own.

Originally published · September 20, 2019

Don’t study Latin if you want to become a better programmer

I used to think that knowledge was strongly transferable. I believed that learning physics could make you a better mechanic. I believed that learning mathematics could make you a better physicist or computer scientist.

After learning a lot of physics and mathematics, I realized that I still found it difficult to write good software, learn about mechanics, design circuits, or understand economics.

This has fundamentally affected my worldview. For example, I no longer take for granted that studying theory can be useful in practice. For me, this is a radical change from my twenties when I believed that computer science wasn’t worth studying since it was just “applied mathematics”.

Since I no longer believe that knowledge is strongly transferable, I have become critical of schooling in general. I used to think that you got smarter with each new college class you took. So I took a lot of them. I took about 30% more classes than necessary to graduate in college. In high school, I took extra mathematics classes outside of the regular schedule (by choice, I don’t even think my parents knew). Thus I know a lot about useless topics.

Some of these classes have turned out to be useful. But less because of the knowledge that they have given me, and more because they have built up my confidence.

For example, all my training in abstract algebra helps me a bit when I want to study random hashing. How much did it help? Well, at some point, I realized that I needed to brush up on Galois fields. I picked up

an undergraduate text, read one chapter, and I was good to go. That is, I knew that I could learn quickly about Galois fields if needed.

However, taking dozen of classes is an expensive way to build up your confidence. A better way would be for you to learn a few difficult things on your own. For example, I hardly know anything about electronics and I never took any class in it, but I know that I could become good at it because I have mastered similar skills.

In any case, because I believe that knowledge is only weakly transferable, I favor learning practical skills that are immediately useful. If you want to become a great software engineer, learn to program better… don’t study Latin.

Given a chance, many parents would cram their kids’ schedule with as many academic classes as possible. The hidden assumption is that kids get “smarter” as they take more classes. But this is almost surely wrong. Of course, there are clear benefits to taking swimming lessons (you learn to swim!) or karate lessons (you learn to fight!), but taking an extra mathematics class might not help as much as you think.

Originally published · April 4, 2014

Why I never give straight answers

There are some questions that you should never answer. This is true when you are facing justice, your angry wife, or students.

Carl Zimmer, a famous science writer, complains about how students keep asking him homework questions. There is no doubt some degree of intellectual laziness mixed with a lack of respect. But I believe that the lazy questions are actually a sign that schools are failing students.

Back when I attended college, you had better have an appointment if you expected to see a professor outside of class. And chances are that the professor would miss your appointment. If you ever got a professor cornered, you had better be prepared. If you came unprepared, with lazy questions, he would dismiss you without a second look. Sending an email to a professor was unthinkable: it would surely go unanswered unless he was your thesis supervisor. (These professors should not be commended for their behavior, of course.)

And yet, experts were far more precious than they are today. There was just no possibility of getting the answers through the Internet. With some luck, you could get a decent textbook at the library, and you had to figure things out by yourself.

Today, students have 10 or 100 times the ressources I had as a college student. Yet hardly any of them have learned to use these ressources. Instead, they email experts and hope for the best.

When I started teaching shortly after graduating, I thought that whenever I got a question by a student, I should answer it as clearly and directly as possible. And so I did. Until I started teaching advanced students, until I started supervising graduate students... and I realized that a whole lot of them were under-prepared for independent work.

They simply could not figure things out on their own. They could not make any judgement call. They couldn't find the right answer by first trying 100 wrong answers.

If I solved a problem in front of them, they could take notes and reproduce my solution. But they often wouldn't even try coming up with an original solution of their own. They certainly would never look critically at the solution I provided.

I realized that we were "over-schooling" the students. Given today's technology, we need to give fewer answers to the students, not more. Though this may sound ironic, I think that students should spend less time in a classroom today than they did in the industrial era.

It is maybe no wonder that employers are not eager to hire college graduates. Beside the bad economy, there might be the very real concern that graduates are ill-prepared to deal with the real world as it is today.

Whenever my kids ask me a question, I almost always ask them back "what do you think?" And then I listen carefully. Often I will ask more questions. Similarly, to teach them something, I often simply ask a question. For example, last weekend, I asked my two boys whether automated teller machines were robots. They agreed that they were robots. Then one of them added "but they lack wheels to move around". And then I asked "but they are still robots?" And they both said "yes, sure". We then launched into a short debate about what a robot is.

I try to follow the same process with my students, including my graduate students. I don't always have the patience. It takes less time and it is more gratifying to simply give the shortest possible answer. But I still try to answer first by a question so that the student is forced to think a little bit about the question. I estimate that 80% of the time, the student ends up answering his own question. I am sure that most students find that I have a bad attitude ("why won't he just give me a straight answer?"), but I am also sure that they also feel proud to have figured it out on their own. And that is what I am ultimately trying to show: you can figure most things out on your own, given a little bit of help.

References

Zimmer, C. (2013, May 31). An open letter to science students and science teachers. *National Geographic Phenomena*. https://phenomena.nationalgeographic.com/2013/05/31/an-open-letter-to-science-students-and-science-teachers/

Wallingford, E. (2013, June). *Actually...* [Blog archive]. University of Northern Iowa. http://www.cs.uni.edu/~wallingf/blog/archives/monthly/2013-06.html

Originally published · June 3, 2013

Elegance as a luxury

As a young student, I was undisciplined. Though I did well in mathematics, I often couldn't be bothered to write down clean computations. I would do the work mostly in my head, and I would quickly jot down the answer. Such a strategy serves you well early on during your education. However, later on, it becomes a handicap.

In part, I blame my schooling. We reward kids that are fast. Who is the best math. student in the class? Whoever can solve the problem fastest. That is entirely wrong. Instead, the best mathematician is whoever can solve the most difficult problems. While speed matters, the ability of scale up with difficulty is much more important. Schools often teach the wrong lesson.

When working on a difficult problem, the problem space often exceeds the capacity of your short-term memory. Though some have better short-term memory than others, the really difficult problems are likely to exceed the memory capacity of any mortal. At that point, method trumps raw speed.

It is useful to think of your brain as a CPU. Modern CPUs are very powerful by historical standards, but they still have very limited memories. What sets apart the best software is how it is able to optimize memory usage so that the CPU does not have to remain idle.

I was never quite able to unlearn my bad habits as a mathematician when working on paper. Part of it is the medium. Though I make extensive use of pen and paper in my daily life, it is mostly for quick collection of simple facts. I never learned to use pen and paper to go deep in my ideas. Instead, at an early age, I learned programming. And it is through a computer that I learned to organize my thoughts.

I am not quite alone in my messiness. Van Emden recently wrote that the famous physicist, Boltzmann, was unapologetically messy as a mathematician. He felt that elegance was a luxury.

But once you see your brain as a CPU, then a more practical question arises. How do you organize the problems to maximize the efficiency of your brain...

I think I am smarter than when I was 22. I can solve more difficult problems more reliably. I am a better software programmer. I am a better mathematician.

- I go slowly. Instead of rushing at the end, I try to make sure I really understand the problem. I try to see its natural components. I divide up the problems in small, manageable parts (that fit in my short-term memory).
  This is where my students most often go wrong. They are trying to go too fast. To try to teach them to go more slowly, I often simply ask them questions about the problem.
- As much as possible, I work locally. That is, I work on parts of the problem that only require a few facts. I only move to another part of the problem once I am done with the current part.
  When I was younger, I would eagerly jump from one part of the problem to another, hoping for a breakthrough. Unconsciously, I was hoping for a flash of insight. I took me years to understand that insights are more likely to come if you focus on a narrow issue.
- I spend a lot of time contemplating alternatives. When I was younger, I would latch on the first credible approach I could think of. Why solve the problem in two different ways when only one way will do? I have since learned that comparing different approaches, and understanding why one is better than the other, is often how you derive the most important insights.
- I spend almost as much time checking my answers than deriving them. I have since learned that even "obvious" facts should be triple checked. I cannot recall how many times I got "stuck" on a problem because of a faulty assumption, or faulty derivation.

References

van Emden, M. (2014, March 8). Boltzmann's brood [Blog post]. https://vanemden.wordpress.com/2014/03/08/boltzmanns-brood/

Originally published · March 10, 2014

Does academic research cause economic growth?

In most developed countries, government massively funds through academic grants, government laboratories, tax credits and research contracts: government R&D alone can often reach 1% of the GDP. In Canada, the government loves tax credits. In the US, the government spends about 60% of all its R&D funding on the military.

Is this government funding a good thing?

What economists tell us is that R&D funding is probably the most important factor determining economic growth. If you want economic growth, then you should be favorable to more research funding. And therefore, it follows that government funding for research ought to be a good idea.

The problem is that it is private R&D that contributes to economic growth, not government R&D:

> The overall rate of return to R&D is very large, perhaps 25 percent as a private return and a total of 65 percent for social returns. However, these returns apply only to privately financed R&D in industry. Returns to many forms of publicly financed R&D are near zero. (Sveikauskas, Bureau of Labor Statistics, Washington, 2007)

> The ubiquitous and fairly pessimistic finding which emerges from the literature is that privately funded R&D contributes significantly to output growth, whereas publicly financed R&D has little or no direct effect. (Capron and van Pottelsberghe, 1998)

> (...) regressions including separate variables for business-performed R&D and that performed by other institutions (mainly public research institutes) suggest that it is the former that drives the positive association between total R&D intensity and output growth. (...) (The Sources of Economic Growth in OECD Countries, 2003).

So the argument that economists have to make to justify government spending on R&D is that when the government pays for research, it entices companies to invest more. Thankfully, the correlation between public spending on R&D and private spending is good, except maybe for government-specific programs (such as the military R&D that the US is so fond of). As a policy, it seems that it makes sense to entice individuals and companies to do more research. Nevertheless, some economists remain cautious:

> (...) the government should be careful in stimulating higher research expenditures. Recent rates of return on R&D are estimated to have reached an all-time low spanning the last 45 years, (...) Firms are also not convinced any more that their R&D investments will yield high returns (...) (Lang, 1999)

Enticing private companies to do more R&D is fine, but are there strong mechanisms turning academic research into more private-sector R&D? There is a correlation, but is there a causal relationship? Most academic research is far removed from economic activities and few professors turn their research into viable products.

Do countries that publish more also see more growth? Yes they do (but wait before concluding):

> The correlation between GDP and publications is (...) high (...) for all countries in the sample analyzed here (77.7-99.6%) (Lee et al., 2011)

This correlation can mean one of two things (or a mix of both):

1. Richer countries can afford more academic research.

2. More academic research makes country richer.

Which is it?

We have some historical evidence in this respect.

In the 17th and 18th centuries, France had the best funded intellectuals in the world. Meanwhile, the British government did not subsidize science and scholarship much. Yet it is in Great Britain that we saw the rise of the Industrial Revolution and of modern science.

Similarly, the US saw a massive economic growth from its early days all the way to WWII… without any public funding for R&D. While other countries like France kept on subsidizing scholarship, it did not put them ahead economically. The US became the dominant world power in 1950 without having invested much at all, as a government, in science and engineering. The US only started investing seriously in research at the end of the 1950s, because it feared the USSR. The USSR was investing in engineering through the state apparatus and the US felt that it had to match it. Remarkably, these massive investments were not followed by an increase in growth. And we know what happened to the USSR.

Of course, today, the US is subsidizing academic research generously: according to some sources, the US is twice as generous as Europe with respect to per capita government R&D. But is the US a wealthy country because of these subsidies, or can it afford these subsidies because its wealth?

Did China get richer because it grew its Universities, or did it grew its Universities because it was richer?

Economists can answer these questions using Granger causality tests. Granger is the economist who showed that we could use the analysis of time series data to establish causality between variables.

And the evidence indicates that, for wealthy countries, government funding for academic research does not **cause** economic growth:

> For European Community member states, the US and Japan correlation between the GDP and number of publications of a given year proved to be non-significant. (…) Studying data referring to consecutive time periods revealed that there is no direct relationship

> between the GDP and information production of countries. (Vinkler, 2008)

> For [rich nations], there is no significant causal relationship between research production and the economy. (Lee et al., 2011)

Conclusion: There are good reasons to pursue academic research. However, academic research is more of a gift economy than an economic growth policy, at least for rich countries. Richer countries can afford to do more academic research, but academic research is not what makes you rich (I should know!).

Disclosure: I have worked in industry R&D, in government R&D and as an academic researcher. I have held a federal research grant for over 10 years. My job would not exist without government funding.

References

Sveikauskas, L. (2007). R&D and productivity growth: A review of the literature. *Monthly Labor Review*. U.S. Bureau of Labor Statistics. http://www.bls.gov/ore/pdf/ec070070.pdf

Capron, H., & van Pottelsberghe de la Potterie, B. (1998). *Assessing the R&D—productivity link: An econometric study on OECD countries and sectors* (STI Working Paper 1998/4). OECD. http://www.oecd.org/sti/inno/1822815.pdf

Organisation for Economic Co-operation and Development. (2003). *The sources of economic growth in OECD countries*. OECD Publishing.

Lang, H. (2009). The changing effect of R&D on productivity growth. *Research Policy*, 38(10), 1574–1582. https://doi.org/10.1016/j.respol.2009.08.001

Lee, S. Y., Kim, B. M., & Lee, S. (2011). Research productivity across OECD countries: A time series analysis. *Scientometrics*, 88(1), 163–181. https://doi.org/10.1007/s11192-011-0476-9

Vinkler, P. (2008). Correlation between the structure of scientific research, scientometric indicators and GDP in 50 and 40 countries. *Scientometrics*, 74(2), 237–254. https://doi.org/10.1007/s11192-008-0215-z

Kealey, T. (2008). *Sex, science and profits: How people evolved to make money*. William Heinemann. https://www.amazon.com/Sex-Science-And-Profits-ebook/dp/B0045JKEQ2/

Pielke, R., Jr. (2013, February 19). Faith-based science policy [Blog post]. https://rogerpielkejr.blogspot.ca/2013/02/faith-based-science-policy.html

Congressional Budget Office. (2007). *Federal support for research and development*. https://www.cbo.gov/sites/default/files/cbofiles/ftpdocs/82xx/doc8221/06-18-research.pdf

Lazowska, E. (2013, February 25). The iPad wouldn't be here without federal research dollars [Interview]. *GeekWire*. https://www.geekwire.com/2013/ed-lazowska-ipad-today-federal-research-dollars/

Johnson, S. (2010). *Where good ideas come from: The natural history of innovation*. Riverhead Books. https://www.amazon.com/Where-Good-Ideas-Come-From/dp/1594485380/

Originally published · February 26, 2013

Science is the belief in the ignorance of experts

Let me put Feynman's quote in context:

> Science alone of all the subjects contains within itself the lesson of the danger of belief in the infallibility of the greatest teachers in the preceding generation (...) When someone says, "Science teaches such and such," he is using the word incorrectly. Science doesn't teach anything; experience teaches it. If they say to you, "Science has shown such and such," you might ask, "How does science show it? How did the scientists find out? How? What? Where?" It should not be "science has shown" but "this experiment, this effect, has shown." And you have as much right as anyone else, upon hearing about the experiments–but be patient and listen to all the evidence–to judge whether a sensible conclusion has been arrived at. (...) The experts who are leading you may be wrong. (...) I think we live in an unscientific age in which almost all the buffeting of communications and television-words, books, and so on-are unscientific. As a result, there is a considerable amount of intellectual tyranny in the name of science.

Many people, including people with a PhD and a career in research, miss Feynman's point. Let me decompose it.

1. How do we know anything? The most common, age-old, approach is to acquire knowledge from a figure of authority or an expert. Your teacher, your mother, your boss. But these experts are routinely wrong, often in unexpected ways:
    - For about thirty years, textbooks said that human beings have 24 pairs of chromosomes (48 in total), even though there were

always 23 pairs. As late as 1955 that was still the consensus. In 1956, Tjio and Levan showed that the correct number is 46. The wrong count had stood roughly from Painter's 1920s estimate until a clear count finally overturned it—not because challenge was impossible, but because authority had settled the question early.

- Before the 1960s, plate tectonics was often ridiculed. Fewer than 20% of geologists accepted the idea of continental drift in the early 1960s, and Pellegrini (2019) tells us that hardly any critic of the idea changed their mind (adopted continental drift). We collectively changed our mind because younger geologists adopted the continental drift theory.
- In 1998, the Nobel Memorial Prize in Economic Sciences recipient Paul Krugman predicted that the Internet would have a small effect ("The growth of the Internet will slow drastically (. . .) By 2005, it will become clear that the Internet's impact on the economy has been no greater than the fax machine's.")
- Heinrich Hertz claimed that radio waves were of no use whatsoever.
- Orville Wright, who literally invented the airplane, said in 1908 that "No airship will ever fly from New York to Paris. That seems to me to be impossible. What limits the flight is the motor. No known motor can run at the requisite speed for four days without stopping, and you can't be sure of finding the proper winds for soaring." On July 6, 1919, a British airship crossed the Atlantic. Lindbergh crossed the Atlantic in an airplane less than 20 years after Orville's statement.
- Scientists held on for years to the idea that space was filled with a substance called the ether even as there was no evidence for it. This belief was held up long after Einstein had proposed the far superior relativity theory. Albert Einstein's relativity was repeatedly mocked and dismissed. Leading members of academia emphatically opposed his special relativity theory. In 1931, a book was published titled *Hundert Autoren gegen Einstein* (*Hundred Authors against Einstein*, Israel et al. 1931)

featuring many arguments seeking to invalidate Einstein's work, despite the existence of multiple experimental validations in favor of Einstein's theory. Without any experimental evidence, the ether would continue to be a popular concept well into the 1930s.

- There was a wide consensus around eugenics in the West at the beginning of the 20th century. Cohen (2016) reminds us: "*Harvard administrators, faculty members, and alumni were at the forefront of American eugenics—founding eugenics organizations, writing academic and popular eugenics articles, and lobbying government to enact eugenics laws. And for many years, scarcely any significant Harvard voices, if any at all, were raised against it.*" Many American States would enact forced sterilization laws on the basis of the scientific consensus around eugenics. Eugenics would motivate some of Hitler's own ideas, leading to the final solution: the extermination of all Jews. Yet, long after the second World War, eugenics would remain popular in the West. The province of Alberta in Canada practiced forced sterilization on the basis of eugenics up until the 1970s. To this day, statistics textbooks celebrate the names of Pearson and Fisher, the leaders of eugenics in the time.

2. Since the beginning of time, we have also given this process of knowledge-from-expertise physical forms. In modern day civilization, we have the peer-reviewed article, the professor at their university, the government scientist in a laboratory coat. In many ways, these people play the same social role as the tribe elder or the priest. The peer-reviewed article is like a sacred text.
3. This pre-existing knowledge and its transmission is often called "science". In such a context, anything can be a science: political science, social science, religious science, and so forth. But whether the knowledge is true or false, it may have little to do with science. It may even be that these institutions that pretend to be about science are unscientific. The fundamental defining characteristic of science, the one that Feynman explicitly identifies, is that we do

not decide whether something is true or false based on authority but rather based on experience. If someone tells you that there are 24 pairs of chromosomes, you have a duty to ask "how do they know?", "how would I find out?".

4. For science... It does not matter whether you are young, old. You can be rich or poor. You can be schooled or not. But you must listen, learn and be patient. In effect, you need to be a constructive skeptic. And you must question your own ideas with even more effort than you question other ideas. Paul Graham puts it well:

> To be a successful scientist it's not enough just to be right. You have to be right when everyone else is wrong. Conventional-minded people can't do that.

Interestingly, once you have appreciated science's true nature as defined by Feynman, you see that it is generally applicable and not limited to Physics. In fact, Feynman clearly believed that the idea of science was applicable to education, for example.

Unfortunately, Feynman was correct. We live in an unscientific civilization. Science survives in niches.

Here are a few unscientific behaviors that are frustratingly common:

1. Teach science as a large set of facts and techniques. I claim that hardly any science at all gets taught in high school. You tell kids that matter is made of atoms which are made of small nucleus surrounded by electrons, at different energy levels. That was taught to one of my sons in high school. Then I asked "how do the scientists know this?" Blank stare. Science is not learning about the charge of the electron by reading a book. If that is all it were, it would be no different than pre-scientific knowledge. No. Science begins when you ask yourself how we can verify what is in the book. How should you teach science? You should essentially teach people to read or listen with constructive skepticism. If you are going to teach science, you must give it in a historical context: you must

stress how our current knowledge is the accumulation of course corrections. You must stress how it is a work in progress.

2. Receive science as a set of facts. We are sometimes given advice and told that it comes from "the science", as if it settled anything. That is often no different than being told that the knowledge comes from "the high priest himself". It is certainly a power play, but it is not science. When someone tells you that science is saying XYZ, you should always ask "How do people know that XYZ is true?" And as an answer, you cannot just be repeating their arguments: you must think for yourself. If you never come up with unexpected questions, you are not a scientist.
3. "I defer to the experts" (said with pride). The defining saying of science should be that nobody knows anything. Deferring to the expert is the defining characteristic of pre-scientific epistemology: doubting the expert is the way of science. Observe that there is a difference between listening to the experts (something Feynman advocated) and deferring to the experts. If your doctor tells you to take some pills, do listen! If you don't understand, do be patient. Don't throw out the expert advice. But don't be ruled by it either!

Science may sound irrational when you spell it out as a doctrine of doubt. If you follow the path of science, you are going to be playing with ideas that are either objectively wrong, or socially wrong, at a much higher rate than if you just followed the experts. But, for scientists, genuine scientists, the goal is not to be right as often as possible. There is no contradiction between "being a good scientist" and "being wrong". The goal is often to find what Peter Thiel might call "secret truths". Everybody knew, at the beginning of the 20th century that it would take decades or centuries before one could fly in an airplane. You could make such a statement with full confidence, even after the Wright brothers had made their first demonstration. And many people of authority were doing just that. To uncover secret truths, you have to train yourself to ask more questions, to carry more doubts.

Rovelli wrote "Not to fear rethinking the world is the power of science". Given that science is fundamentally subversive, how could

it emerge and survive? I believe that scientific thinking is part of a broader ideology that gives its bearers an evolutionary edge. Simply put, societies that make room for science have an edge. They build and deploy better technology. They adapt more quickly to change.

References

Feynman, R. P. (1969). What is science? *The Physics Teacher*, 7(6), 313–320.

Pellegrini, A. (2019). Continental drift acceptance: A case study in the sociology of science. *Journal for General Philosophy of Science*, 50(1), 9–31. https://doi.org/10.1007/s10838-018-9439-7

Mikkelson, D. (2013, October 16). Did Paul Krugman say the Internet's effect on the economy is no greater than the fax machine's? *Snopes*. https://www.snopes.com/fact-check/paul-krugman-internets-effect-economy/

PCMag Staff. (n.d.). Six things you didn't know about Heinrich Hertz. *PCMag*. https://www.pcmag.com/news/six-things-you-didnt-know-about-heinrich-hertz

Smithsonian Education. (n.d.). The Wright brothers: Flights of the future. https://www.smithsonianeducation.org/educators/lesson_plans/wright/flights_future.html

Israel, H., et al. (1931). *Hundert Autoren gegen Einstein*. R. Voigtländer.

Cohen, A. (2016, March–April). Harvard's eugenics era. *Harvard Magazine*. https://www.harvardmagazine.com/2016/02/harvards-eugenics-era

Goldman, W. (1983). *Adventures in the screen trade: A personal view of Hollywood and screenwriting*. Warner Books.

Rovelli, C. (2021). *Helgoland: Making sense of the quantum revolution*. Riverhead Books. https://www.amazon.ca/Helgoland-Making-Sense-Quantum-Revolution/dp/0593328884/

Originally published · July 12, 2020

Enough with the bogus medical studies!

Every week, we hear about how eating such and such food gives cancer, or how working out can save you from a heart attack. If you have been reading these studies with attention and changing your life to follow their recommendations… you have been wasting your time and you have possibly harmed yourself.

Feynman famously described a common phenomenon which he called cargo-cult science. True science is a fantastic tool that has taken us, in a couple of centuries, from superstitious fools to a highly advanced civilization. Almost all major advances in the last two centuries can be linked with science. So people want to appear to be doing good science… but they are much less concerned about actually doing good science, when they even know what it is.

What is science? As Feynman wrote, it is the belief in the ignorance of experts. It is the idea that our intuition, our good sense… are not enough to tackle the world. We need objective measures. We often need mathematics. We often need hard logic. We need repeatable experiments capable of falsifying theories.

We need these things because we are easily fooled. Science acts as some sort of "software update" for the brain. You train your brain to rely on objective, verifiable facts, and hard logic… instead of impressions.

One of the core ideas behind science is that authority-based arguments should be rejected. So if Professor X from Harvard University or MIT says something… it is not scientific to believe her because she is from Harvard, or because she is a professor. It is similarly unscientific to believe something because it has appeared in a prestigious journal.

It is not even scientific to believe something because of a "consensus". (Hint: there was a widespread consensus that the airplane was "impossible" before the Wright brothers came along.)

Let me take this further. Believing something because impressive people say it must be so is the illustration of pre-scientific thought. And this means that every time you hear that a study by impressive people proved something new, you should doubt… you should doubt automatically, without even thinking.

How do we determine what is true? That's one of the core lessons of the scientific revolution: it is much harder than you might expect. We are feeble creatures that very easily fall prey to bogus arguments. We must relentlessly seek incontrovertible evidence. There is no shortcut.

The real question is not how to make your mind about what is true… that is easy, your brain will eagerly accept many things as true given a chance. The hard part is to keep on doubting. We are just not wired for constant doubt, we are "believers".

Many people do not really understand what science is… and this includes fully trained "scientists" who receive government research grants and train other scientists. What they do understand, however, is how to speak and act like a scientist. It looks like science, but it does not give the same result. That's what Feynman calls cargo-cult science.

When it comes to medicine, we invest a lot of money in medical research. Several times more than any other type of research. Secondly, lots of real people can end up suffering needlessly if the research is wrong.

Is medical research wrong? Yes. But that's not the most tragic part. The real problem is that much of it is pure cargo-cult science. And well funded at that. Cargo-cult science does not merely lead to the wrong ideas… it actively sets us back.

What are the problems?

Most work is based on observational studies

Just watching and taking notes is not science. It is valid scholarship, as long as you report it as such. But it gets reported as "science" instead which is misleading.

There is nothing wrong with historians who tell us that Vikings once traveled to North America. It is interesting. It is scholarship. But it is not science. It is an observation.

Don't get me wrong. Scholarship is precious. We know, for example, that women live longer than men. Cataloging such facts can help us… but it is not science. Our ancestors knew exactly where the stars were. They knew a lot about the seasons. Their high priests observed, cataloged… but they did not know about science.

When you hear that exercise has been shown to reduce your risk of cancer… that's (usually) a flat out lie. What actually happens is that scientists take a group of people. Then they ask them whether they exercise. Then they ask whether they have cancer. And then they run the statistics. "Oh! See, the people who say they exercise also less often say that they have cancer." So far so good, but you have merely observed, you have not done any science. You have done nothing that is above what scholars routinely did in the pre-scientific era.

All you can show with observational studies is an observation. Nothing else. It can be a lot, but it is not science.

It could be that when Joe complains about his knee, it rains. That does not show that complaining about one's knee brings the rain. Nor does it show that the rain brings pain to one's knee. It is an observation, not science.

Small effects are greatly exaggerated

Even when scientists actually do science… they do not use the proper tools to avoid being fooled or fooling others. In fact, they often appear to seek to fool us.

Suppose that you are treating patients for cancer. The average survival rate is 9 months. All patients die within 3 years. You give them some pill and the survival rate increases to 10 months, but all patients still die within 3 years. It is progress, but a small progress. If the pill costs $100,000 and makes you twist in pain for 3 weeks, maybe it is no progress at all.

However, it is not how most studies will present it. They will choose a short window. Say a year. Maybe 60% of all patients die within a year. However, with the extra month offered by the pill, only 40% of

all patients die within the year. Then the study will conclude that the survival rate has been increased by 50%. I kid you not.

They report selective effects

Without even looking at the numbers, I can tell you that people with high-risk occupations have a lower cancer mortality. Why? Because they die younger. So, there you go… I got a cancer prevention technique: choose a high-risk occupation. Of course, this reasoning is silly. But that's often how medical studies are presented.

So how to keep cancer free? You will hear things like "eating well, exercise, keeping your weight low, eating broccoli, avoiding red meat, getting tested repeatedly". But if you actually look at the numbers, mostly, there are two factors that have an effect worth noticing on your risk of having cancer: your age and whether you smoke. Your risk of having cancer grows up exponentially with age. If eating bacon increases your cancer risk, it is by a tiny amount. What about early diagnostic? Again, a nice idea, but for many popular screening programs the benefits are smaller—or more ambiguous—than the headlines suggest, once you account for false positives, overdiagnosis, and who is being screened.

What about heart attacks? Age and gender are basically the things you need to watch, assuming you are not smoking. Again, your risk grows up exponentially with age. Also, men are three times more likely to have a heart attack than women. Why? We don't know. Smoking is often considered like a really bad habit. It is true that it doubles your risk of cardiovascular diseases. However, aging from 40 to 90 multiplies your risk of cardiovascular diseases by 100.

So, how do you stay healthy? By staying young. Sadly, we don't know how to prevent aging. Yet.

Demonizing hamburgers is a dangerous distraction.

Note: I am not a medical professional. Consult your doctor if you have questions. While you are at it, ask her if she knows about cargo-cult science.

References

Feynman, R. P. (1974). Cargo cult science. *Engineering and Science*, 37(7), 10–13.

Originally published · May 30, 2016

The Future Is Malleable

These thirteen essays—and a short interlude—make the case for techno-optimism—not as wishful thinking, but as a conclusion drawn from decades of watching hard problems yield to patient effort. They extend that argument into the present: scientific institutions, artificial intelligence, models of innovation, and what the next wave of tools might—and might not—change about how we work and discover.

The case for techno-optimism

When I wrote this in 2015, voice search on a phone felt like a small miracle. A decade later, large language models draft code, summarize papers, and answer questions my sons used to ask Google. The examples below have aged, but the pattern has not: progress arrives unevenly, often invisibly, and pessimists mistake the lag in clinics and classrooms for stagnation in the underlying science.

I define techno-optimism as the belief that technology makes us healthier, richer and smarter at an accelerated rate.

Anyone working in information technology cannot help but be a bit of a techno-optimist. My wife's latest Samsung phone is a technological marvel that can replace dozens of expensive devices from ten years ago (phone, camera, PDA, ...). Google's voice recognition software has finally become good enough. Here is what I did just now:

– Ok Google

– Where is Denmark?

And it showed me a map of Denmark. Keep in mind that I have a thick French accent.

My 9-year-old son comes by (his English is even more approximate) and he says "Ok Google. Who is Mario?". And sure enough, it worked.

A common objection to techno-optimism is that it only works in information technology. For example, there seems to be a widespread belief that medicine or education is standing still.

It is true that if you go in your doctor's office, it might feel like traveling back in time to the 1970s. But let us keep in mind that, in the future, most things will still look the same. Except for my wicked flat TV, my living room would not surprise someone from the 1950s. A wall is a wall: we just made it much cheaper and faster to build walls. A desk is a desk: they have just become much cheaper. Paper is

still very useful: we have just learned how to cost-efficiently replenish our forests.

Moreover, it will always be true that the future won't be uniformly distributed. Even today, one out of eight people never use the Internet. This cannot be helped. Progress never arrives everywhere at once.

But really, medical progress is fast and furious:

- A paraplegic woman, Jan Scheuermann, can feed herself using a robotic arm controlled by her brain. This has been commonly done in monkeys for many years.
- There is now a medical specialty called resurrection. In forward-thinking hospitals, "dead" patients have their blood artificially oxygenated. In this manner, patients can remain dead for hours before being brought back.
- We can restore partial sight to the blind using retinal implants. Researchers expect to start human trials in the next few years using a stem cell therapy to repair retinal damage.
- AIDS is a devastating but "manageable disease". Infected people must take expensive and harsh drugs. But it looks like we might finally defeat AIDS using gene therapy. We found out that some people were naturally immune to the AIDS virus. We identified the gene responsible, and we simply edit the corresponding gene in the cells of infected people. This actually works in real people. Moreover, at least one patient has been cured of AIDS (back in 2008) thanks to stem cell transplants.

In education, progress is equally amazing. Let us not forget that the web itself is probably the greatest learning tool ever invented. It is orders of magnitude superior to previous alternatives. For example, many of my colleagues have substantially improved their English through online services. They are both more effective and much cheaper than human tutors. To help my kids with their spelling, I have created a set of small web pages. The scripts in the web pages speak out the words (in French!) and my sons have to write the words correctly. This alone would have sounded like science fiction ten years ago: a little script in a web page can speak out words in any language! Yes, what actually

happens in the classroom might be stuck in time… but progress is everywhere. You just have to know how to look.

And that is what it comes down to. Progress happens at first mostly in pockets. It then propagates at an uneven pace. But eventually, it touches nearly everyone… in ways that we often do not notice until later if at all.

That is why techno-optimism is not a common stance despite the overwhelming evidence… good technology is nearly invisible. So people greatly underestimate what can be done given a few decades.

References

Collinger, J. L., et al. (2013). High-performance neuroprosthetic control by an individual with tetraplegia. *The Lancet*, 381(9866), 557–564. https://doi.org/10.1016/S0140-6736(12)61816-9

Parnia, S. (2013, April 6). The Lazarus effect: The science that is rewriting the boundaries between life and death. *The Guardian*. https://www.theguardian.com/society/2013/apr/06/sam-parnia-resurrection-lazarus-effect

Dobelle, W. H. (2000). Artificial vision for the blind by connecting a television camera to the visual cortex. *ASAIO Journal*, 46(1), 3–9.

Tebas, P., et al. (2014). Gene editing of CCR5 in autologous CD4 T cells of persons infected with HIV. *The New England Journal of Medicine*, 370(10), 901–910. https://doi.org/10.1056/NEJMoa1300662

Allers, K., et al. (2011). Evidence for the cure of HIV infection by CCR5-delta32/delta32 stem cell transplantation. *Blood*, 117(10), 2791–2799. https://doi.org/10.1182/blood-2010-09-309591

Originally published · June 26, 2015

Technology sets the bar higher

I am old enough that, as a kid, I did not have access to a calculator. My mother, a teacher, had an electronic calculator that you had to plug in the wall. She would use it to crunch in the grades at the end of each term. It felt fantastically modern.

In any case, one night, when my mother was away, I decided to slide into my mother's bedroom, plug the calculator and use it to do my math homework. I was in third grade. This was, let us be clear, cheating.

What happened? I got a terrible grade. Turns out that using a calculator does not guarantee the correct results.

I lived through a similar experience when I got to college. After learning algebra by hand for years, I soon realized that computers could do algebra too! I had discovered computer algebra systems. Wow! So mathematics would be easy from now on. But as I would soon learn, these systems allowed evil professors to ask even harder problems...

In 2009, for the proof of a result, I had a computer algebra system (in this case Maxima) run a long script to check many possible cases. This intensive search would have taken days or weeks to do by hand. Yet I was able to do this work with the help of my computer precisely because I have some degree of mathematical sophistication... A naive user would not have known what to ask of the computer. The better at mathematics you are, the more you can get out of computer algebra systems...

The Guardian recently reported that some expect that within 25 years, we will get instant language translation. Contrast this with the fact that Richard Stallman, one of the world's leading hacker and free software advocate, is fluent enough in French to give talks. In fact, I

find that Stallman sounds more dignified and polite in French, but maybe it is just me.

Is learning a foreign language a waste of time given that computers can translate?

Let me consider programming languages as an analogy. Most programming languages today are Turing complete. And all Turing complete programming languages are mathematically equivalent. So you only ever need to learn one programming language… they are all equivalent! But this is abstract nonsense as any programmer will tell you. And it is just as much nonsense to claim that computer translation makes learning a foreign language obsolete.

As an aside, reputed AI experts in the 70s and 80s were discouraging people from learning to program. They expected that very high level languages would make programmers obsolete by 2000 or 2010. I do not think they could have been more wrong. Who benefited most from software technology? People who learned to program.

Or consider spelling. Should I give up teaching my boys to spell flawlessly given that we all have spelling autocorrection? I would say that spelling and grammar checkers have, if anything, raised the bar. Kids today need to master more thoroughly the grammar, and they need to spend more time studying the nuances of spelling.

The problem is the same whether we use calculators, computer algebra systems and language translation… You need to be smart to use technology effectively. As we acquire more and more technology, we need to get smarter.

It is true that as technology evolves, we should learn things differently. For example, there is little need to train students in very technical algebra. Much of the standard curriculum today is antiquated. Last night, I was doing a word problem with one of my boys and he got discouraged because he had to compute 70% of 280. I told him that I would never expect him to do this by hand… what mattered to me is that he was smart enough to realize that the answer was 0.7 times 280. Similarly, when learning a foreign language, you should probably focus more on how the languages differ, than on vocabulary and

spelling. But one thing is certain: with increased technology comes higher intellectual requirements.

Originally published · March 28, 2014

The courage to face what we do not understand

Sadly, it is easy to forget that what we know is all but a tiny fraction of all there is to know. Human beings naturally focus on what they understand. The more you learn, the stronger this phenomenon tends to be.

Irrespective of any biological mechanism, I believe that it is a form of "aging" in the sense that it makes you increasingly inflexible. The more you know, the less open you are to new experiences. In effect, the dumber you get.

Let us call this "cognitive rigidity".

This almost seems unavoidable, doesn't it?

I believe that there are at least three factors driving cognitive rigidity:

- Economics often favors specialization. Suppose that you have been programming in Java for the last 5 years. You have an enormous advantage over anyone who starts out in Java. So you have every incentive to ignore new programming languages.
- Skills and experience are often poorly transferable. For example, I speak fluently in French and English. If I try to learn Chinese, it will take me years of hard work just to catch up with a 6-year-old child born in China.
- The more and longer you focus, the more likely you are to hit diminishing returns. There is very little difference between spending 5 years programming Java and spending 20 years doing the same.

What is interesting is that cognitive rigidity is an entirely general process.

For example, there is no reason to believe that an artificial intelligence would not suffer from cognitive rigidity. Suppose that you train a machine for some task over many years. The machine has gotten very good at it, and any small change is likely to make it worse at its job. At some point, the machine will stop learning. It has fallen into a local extremum. Yet it is possible for a whole other piece of software to come in and surpass the old machine because it starts from new assumptions. The old machine could have explored new assumptions, but that would have likely provided no gain.

Organizations, communities, and even whole countries might also be subject to cognitive rigidity. For example, IBM famously missed the PC revolution… then Microsoft nearly missed the Internet revolution, and squarely missed the mobile revolution.

Where we should be really concerned is that I believe humanity as a whole can fall victim to cognitive rigidity. I have recently "reinvented" myself as an advocate for techno-optimism. I did so when I realized that even among people who should know better, there was a massive failure of imagination. Even young computer scientists fall for it. I find that people universally imagine the future as the present, with a few more gadgets. Though I do not share the pessimism, it is not what troubles me. What troubles me is that people assume we have reached worthwhile extrema. We haven't!

- We do not know what intelligence is. We can emulate some of the human intelligence in computers. We can "measure" intelligence using IQ tests… but we do not know what it is. Not really. We can't even reproduce the visual recognition abilities of a rodent, despite the fact that we have more than enough CPU cycles to do it. In a very fundamental way, we do not know anything about intelligence.
- We are nowhere close to figuring out the laws of the universe. At a high level, we have two systems, quantum mechanics and relativity, that we glue together somehow. It is an ugly hack.
- We really do not know much about biology. We have had the (nearly) complete human genome for many years now. So we can

touch the binary code of life. We can change it. We can tune it. But we have no idea how it works. Not really. We don't know why we age. We don't know why we get cancers while other animals don't. To make matters worse, there is a hidden form of cognitive rigidity when people consider biology. There is a strong assumption that whatever natural evolution produced, it must be ideal... and so tinkering with it is dangerous. For example, I was telling my neighbor about the existence of genes that make people stronger, or more resilient to cancer. His first reaction was that these genes must come at a sinister cost, otherwise we would all have them. This is, of course, a fallacy. You could equally say that whatever product has not yet been marketed must not be profitable, otherwise someone else would have already marketed it.

- Our best practices in politics and economics are based on debatable heuristics that work "ok", but they are probably nowhere close to being optimal. Alastair Reynolds in his novel Revelation Space depicts a highly advanced human society where people have adopted radically different forms of politics and economics. His novel hypothesizes that this led to a surge of prosperity never seen before. Yet almost any debate that puts into question current politics is a non-starter. People simply assume that whatever they have must be the best that can be had.

So we have these giant gaps. What really worries me is how most people do not even see these huge gaps in our knowledge. And these are just the beginning of a long list. If you drill down on any given issue, you find that we know nothing, and we often know less than nothing.

I do not think that cognitive rigidity is unavoidable. I don't think that there is a law that says that you have to fall prey to it. For example, while Intel has had every occasion to fall prey the way IBM and Microsoft did, it has time and time again been able to adopt new techniques. We still look at Intel today to determine whether Moore's law is holding, 40 years after the law was written.

The key to progress is to have the courage to face what we do not understand. But it takes courage to face the unknown.

References

Reynolds, A. (2000). *Revelation space*. Gollancz.

Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, 38(8), 114–117.

Originally published · December 22, 2015

Beyond papers

For the last few decades, we have defined science by the publication of peer-reviewed research. That is, a scientist is someone who can write a paper and have it reviewed by two or four other scientists who conclude that the work is credible.

Notice how this is a circular definition: it defines a scientist as, effectively, a member of a social club of people who agree on writing conventions.

Contrary to what many people believe, this system of peer review is recent. When Watson and Crick submitted their 'double helix' paper to *Nature* in 1953, it did not go to reviewers.

Peer review became the dominant model in the 1970s. It emerged around the same time the West adopted aspects of the Soviet model of science. It also coincided with a decade-long stagnation in science. I believe it was a mistake.

Sure, science is still making amazing progress. I celebrate the incredible scientific productivity of enterprises like Google DeepMind and OpenAI—enterprises that do not rely on conventional peer review.

This model of 'peer-review science' now faces an existential threat. In fact, I do not believe that the current dominant model of science, established in the early 1970s, can survive. Indeed, Large Language Models can produce papers today that can pass the peer review test. In a sense, it is an even easier test to pass than the Turing test, because the conventions of peer review science are so artificial.

Why would anyone be paid to write papers that are indistinguishable from what ChatGPT can produce in mere seconds?

So where do we go from here? There is only one option. We have to stop considering the research paper as the end product of research.

It never was, in any case. That was always a convenient bureaucratic convention, inspired by the need to 'manage science.'

If ChatGPT can produce research papers that are indistinguishable from what most scientists can write, then maybe scientists can focus on actually advancing science—something that ChatGPT has thus far proven unable to do.

As I have argued elsewhere (Lemire, 2024), AI's ability to generate vast amounts of text raises concerns about a potential flood of irrelevant theoretical papers, further straining the evaluation system. Stonebraker's (2018) call for rewarding problem-solving over publication needs revisiting. Perhaps the emphasis should be on the impact and significance of research, not just its passage through peer review—a skill replicable by AI.

References

Stonebraker, M. (2018). How to build a really good database system. In *CIDR 2018, Conference on Innovative Data Systems Research*. https://www.cidrdb.org/cidr2018/papers/p49-stonebraker.pdf

Lemire, D. (2024). Will AI flood us with irrelevant papers? *Communications of the ACM*, 67(9), 9. https://doi.org/10.1145/3673649

Originally published · October 3, 2025

Will emerging artificial intelligence boost research productivity?

Bryan Caplan, an economist, raised an interesting question on Twitter: why aren't people celebrating the fact that tools like GPT might soon allow us to produce many more research papers at the same cost, without sacrificing quality?

The American government is funding research to the tune of 138 billion dollars a year. It includes 45 billion dollars solely for the National Institutes of Health. There are governmental laboratories, special projects, the National Science Foundation, and so forth. And, of course, you have military research. For comparison, a large corporation like Google, with hundreds of thousands of employees, makes about 20 billion dollars in profit each year. We pour a significant effort into scientific research.

The number of research articles published per year follows an exponential curve. We publish millions of research articles. We never published so many research papers, and we will publish many more next year. Thousands of researchers publish at least a research paper a week.

At a glance, the objective of much of this research is to produce research articles. It seems certain that if we could produce more of them for the same cost, the national research productivity would go up.

The problem, as pointed out by Caplan, is that nobody cares about these research articles. In fact, if you asked engineers to pay the

authors to access their research articles, they almost certainly would not be willing to pay.

To be clear, Caplan does not mean 'all of it'. Some of the work is widely read and influential. He estimates that the worthwhile publications make up 2%. Nevertheless, most published articles are not contributing anything at all. So our ability to write more of them will not help us in any way.

As people like Stonebraker have remarked, we have lost our way: people publish to get jobs and promotions, and no longer to advance science. The focus on peer-reviewed publications has more to do with a status competition between researchers and academics, than a genuine desire to advance science. The metrics have been gamed. Researchers, generally, no longer have customers: nobody uses the result of the work, except maybe other researchers. Peer-reviewed papers are getting increasingly boring.

It does not mean that actual scientific progress does not happen. Nor does it mean that we should stop writing papers and books. The research done at OpenAI is changing the world, and they are writing research papers. However, if you go on OpenAI's web site, under Research, you find a list of links to arXiv, a freely accessible repository of papers that are often not formally peer reviewed. In other words, the researchers at OpenAI seem more concerned with advancing the science than with playing a "publication game".

Effectively, we have a severe misalignment problem. What gets you a job, a promotion or a research grant, no longer has much to do with whether you are advancing science.

Sometimes, when I sit on a recruiting committee for new professors, I ask the candidate to tell us about their most important scientific contribution. I then ask them why it is an important contribution. Almost invariably, some people will say "it was published in this important conference". A large fraction of academia is thoroughly confused about what constitutes progress.

What will tools like GPT do to the business of science? It is hard to tell. However, it is possible that the net outcome could be greatly positive. If "publishing a paper" is no longer a significant achievement

thanks to artificial intelligence, it is possible that it could stop being a status-seeking game. This would leave the field to people who seek to contribute real advances.

I think that it is conceivable that this new breakthrough in artificial intelligence could bring us to a new golden age of scientific research. Of course, people will still seek to game prestige signals, but a fast changing technological landscape could reset the metrics and re-align the incentives.

References

Stonebraker, M., Madden, S., Abadi, D., Harizopoulos, S., Hachem, N., & Helland, P. (2007). The end of an architectural era: (It's time for a complete rewrite). In *Proceedings of the 33rd International Conference on Very Large Data Bases* (pp. 1150–1160).

National Science Foundation. (2022). *National patterns of R&D resources*. U.S. National Science Foundation. https://ncses.nsf.gov/pubs/nsf24330

Bornmann, L., & Mutz, R. (2015). Growth rates of modern science: A latent piecewise growth curve approach to model publication numbers from established and new literature databases. *Humanities and Social Sciences Communications*, 2, 170. https://doi.org/10.1057/hsscomms2015.62

Caplan, B. ([@bryan_caplan]). (2023). Posts on AI and research productivity [Microblog posts]. *X*. https://x.com/bryan_caplan

Originally published · April 21, 2023

Embodied cognition and agentic AI

Where is your intelligence located? In your brain?

It is a simplistic answer. A better model is that your intelligence is embodied.

Consider a cook working at an expensive restaurant. He has all his favorite knives and cooking instructions, placed exactly where he wants them. His kitchen is part of his intelligence, of his skills. The same cook working in your kitchen can probably cook better than you do, but he can't reproduce the same meals he would prepare in his favorite kitchen.

We often assess computer programmers using whiteboard tests. It is an endless source of complaints. Programmers rightly point out that it forces them out of their element. They are just not as good when you take away their laptop. It is not an excuse, it is a real issue: you are cutting them off from part of what makes them so intelligent.

To sum it up, the model of intelligence as a brain in a jar, disconnected from anything else, is ridiculous.

If you accept the idea of embodied intelligence, then many actions that we view as a consequence of our intelligence are actually part of our intelligence. First and foremost, language. Our ability to talk or write to each other means that I am not limited by my own person. Have you ever heard of human beings isolated in small tribes making technological breakthroughs? Nah. Progress requires lots of people communicating together. Up until a few decades ago, progress required cities. Today I am less certain that it does, as I can more and more communicate with anyone in the world from anywhere. But language is still critical, we have not invented anything better. Similarly, having

hands and the ability to build sophisticated tools (like laptops) allows us to extend our intelligence.

At the end of 2022, we got a breakthrough technology: ChatGPT. It built on several pre-existing ideas such as (large) language models, neural networks, and so forth. That's the 'GPT' part. But an important, if underappreciated, part of the breakthrough was the 'Chat' component. Someone had the idea of connecting a large language model with a chat interface. Maybe this came naturally and obviously to people building this system, but it should not be assumed to be trivial or unimportant.

Language is a key component of our intelligence, and, thus, it makes sense that it would be pivotal for machine intelligence.

We embodied the AI software in a chat box.

The next step was what we call today 'agentic AI'. We keep the chat box, but we add the ability for the AI software to interact with tools, and to make plans to use them. In effect, we give the AI more agency: it can do stuff and learn from the results as they happen. It is starting to resemble a human being with hands and tools.

I was talking with a colleague this week. My colleague is all in on the AI revolution. He uses his AI to help him write better and faster, and to get his data analysis done faster, without so much help from technical experts.

But my colleague was not aware of the agentic AI approach. I tried to explain on the phone. What does it mean to give the AI access to tools? Is this only about saving the effort of copying and pasting the AI's response?

To illustrate the idea, I started an AI in a shell within something called RStudio. It is an environment people use to program in R, to do data analysis. I don't use R or RStudio, but thanks to the AI, I was able to build an entire climate research project in a few minutes, complete with the retrieval of the data from the web.

How did the AI do it? It tried a few things, initially struggling to download the data. At some point, it finds out that it needs new R packages, so it installs them, and once they are installed, it can proceed to generate figures, verifying that it works.

Agentic AI greatly extends machine intelligence by improving the embodiment of AI.

I believe that it is not yet understood as it should be.

In Montreal, the most established professor in the field of AI is Yoshua Bengio. He started his own non-trivial enterprise a few years ago (Element AI). His latest venture is Law Zero, which aims to create a Scientist AI. The first goal of this project is to build AI without the agentic component. It should be a disembodied AI that has no goal of its own, no agency.

I fear that Bengio suffers from what Kevin Kelly called Thinkism. Let me quote from Kelly's 2008 essay.

> No intelligence, no matter how super duper, can figure out how human body works simply by reading all the known scientific literature in the world and then contemplating it. No super AI can simply think about all the current and past nuclear fission experiments and then come up with working nuclear fusion in a day. Between not knowing how things work and knowing how they work is a lot more than thinkism. There are tons of experiments in the real world which yields tons and tons of data that will be required to form the correct working hypothesis. Thinking about the potential data will not yield the correct data. Thinking is only part of science; maybe even a small part. (...) Thinkism is not enough. Without conducting experiments, building prototypes, having failures, and engaging in reality, an intelligence can have thoughts but not results. It cannot think its way to solving the world's problems. (...) The Singularity is an illusion that will be constantly retreating — always "near" but never arriving. We'll wonder why it never came after we got AI. Then one day in the future, we'll realize it already happened. The super AI came, and all the things we thought it would bring instantly — personal nanotechnology, brain upgrades, immortality — did not come. Instead other benefits accrued, which we did not anticipate, and took long to appreciate. Since we did not see them coming, we look back and say, yes, that was the Singularity.

I believe that university professors are especially prone to thinkism. They view intelligence as being centered on what is happening in their brain. When you live in an ivory tower, it is easy to dismiss the real world as the core source of intelligence. Further, they are often people who did quite well in school where thinkism is naturally prevalent.

I have been a professor most of my life. However, I tire quickly of talking with other professors. What I most enjoy is working with people who have new tools that they apply in the real world. Unsurprisingly, I spent most of my time working with software that people deploy in the real world.

What Kelly is saying is that a high degree of intelligence is not enough to do much of anything. The real world is not the final stage of your thinking process. It is maybe the most important part of it.

And thus, when you connect your AI with the real world, giving it the ability to run experiments (as virtually all software developers do today), you get impressive results that go much beyond what AI software can do on its own.

Agency is not a feature. Agency is primary.

References

Kelly, K. (2008, January 3). Thinkism. *The Technium*. https://kk.org/thetechnium/thinkism/

Bengio, Y. (2025). *Law Zero* [Company announcement]. Law Zero. https://lawzero.ai/

Originally published · May 28, 2026

Interlude

How helpful is AI?

Do large language models (AI) make you 3x faster or only 3% faster? The answer depends on the quality of the work you are producing.

If you need something like a stock photo but not much beyond that, AI can make you 10x faster.

If you need a picture taken at the right time of the right person, AI doesn't help you much.

If you need a piece of software that an intern could have written, AI can do it 10x faster than the intern.

If you need a piece of software that only 10 engineers in the country can understand, AI doesn't help you much.

The effect is predictable: finding work if your skill level is low becomes more difficult. However, if you are a highly skilled individual, you can eliminate much of the boilerplate work and focus on what matters. Thus, elite people are going to become even more productive.

Originally published · April 7, 2025

Antifragile programming and why AI won't steal your job

Whenever I say I dislike debugging and organize my programming habits around avoiding it, there is always pushback: "You must not use a good debugger."

To summarize my view: I want my software to be antifragile (Taleb, 2012). The longer I work on a codebase, the easier it should become to fix bugs.

Each addition to a piece of code can be viewed as a stress. If nothing is done, the code gets slightly worse, harder to maintain, more prone to bugs. Thankfully, you can avoid such an outcome.

That's not natural. For most developers lacking deep expertise, as the codebase grows, bugs become harder to fix: you chase symptoms through layers of code, hunt heisenbugs that vanish in the debugger, or fix one bug only to create another. The more code you have, the worse it gets. Such code is fragile. Adding a new feature risks breaking old, seemingly unrelated parts.

In my view, the inability to produce antifragile code explains the extreme power-law distribution in programming: most of the code we rely on daily was written by a tiny fraction of all programmers who have mastered antifragility.

How do you reverse this? How do you ensure that the longer you work on the code, the shallower the bugs become? There are well-known techniques, and adding lots of tests and checks definitely helps. You can write antifragile code without tests or debug-time checks… but you'll need something functionally equivalent.

Far-reaching prescriptions ("you must use language X, tool Y, method Z") are usually cargo-cult nonsense. Copying Linus Torvalds'

tools or swearing style won't guarantee success. But antifragility is not a prescription, it is a desired outcome.

Defensive programming itself is uncontroversial—yet it wasn't common in the 1980s and still isn't the default for many today. Of course, a full defensive approach isn't always applicable or worth the cost.

For example, if I'm vibe-coding a quick web app with more JavaScript than I care to read, I'll just run it in the browser's debugger. It works fine. I'm not using that code to control a pacemaker, and I'm not expecting to be woken up at midnight on Christmas to fix it.

If your program is 500 lines and you'll run it 20 times a year, antifragility isn't worth pursuing.

Large language models can generate defensive code, but if you've never written defensively yourself and you learn to program primarily with AI assistance, your software will probably remain fragile.

You can add code quickly, but the more you add, the bigger your problems become.

That's the crux of the matter. Writing code was never the hard part—I could write code at 12, and countless 12-year-olds today can write simple games and apps. In the same way, a 12-year-old can build a doghouse with a hammer, nails, and wood. Getting 80% of the way has always been easy.

Scaling complexity without everything collapsing—that's the hard part.

References

Taleb, N. N. (2012). *Antifragile: Things that gain from disorder*. Random House.

Originally published · November 29, 2025

AI: igniting the spark to end stagnation

Much of the West has been economically stagnant. Countries like Canada have failed to improve their productivity and standard of living as of late. In Canada, there has been no progress in Canadian living standards as measured by per-person GDP over the past five years. It is hard to overstate how anomalous this is. Late-Soviet growth had already slowed into a grinding stagnation that people treated as a political emergency; a rich Western country flatlining living standards for years should feel like the same kind of emergency.

Late in 2022, some of us got access to a technical breakthrough: AI. In three years, it has become part of our lives. Nearly all students use AI to do research or write essays.

Dallas Fed economists projected the most credible effect that AI might have on our economies: AI should help reverse the post-2008 slowdown and deliver higher living standards in line with historical technological progress.

It will imply a profound, rapid but gradual transformation of our economy. There will still be teachers, accountants, and even translators in the future… but their work will change as it has changed in the past. Accountants do far less arithmetic today; that part of their work has been replaced by software. Even more of their work is about to be replaced by software, thus improving their productivity further. We will still have teachers, but all our kids, including the poorest ones, will have dedicated always-on tutors: this will not just be available in Canada or the USA, but everywhere. It is up to us to decide who is allowed to build this technology.

AI empowers the individual. An entrepreneur with a small team can get faster access to quality advice, copywriting, and so forth. Artists with an imagination can create more with fewer constraints.

I don't have to prove these facts: they are fast becoming obvious to the whole world.

New jobs are created. Students of mine work as AI specialists. One of them helps build software providing AI assistance to pharmacists. One of my sons is an AI engineer. These are great jobs.

The conventional explanation for Canada's stagnation is essentially that we have already harvested all the innovation we are ever going to get. The low-hanging fruit has been picked. Further progress has become inherently difficult because we are already so advanced; there is simply not much room left to improve. In this view, there is no need to rethink our institutions. Yet a sufficiently large breakthrough compels us to reconsider where we stand and what is still possible. It forces us to use our imagination again. It helps renew the culture.

We often hear claims that artificial intelligence will consume vast amounts of energy and water in the coming years. It is true that data centers, which host AI workloads along with many other computing tasks, rely on water for cooling.

But let's look at the actual water numbers. In 2023, U.S. data centers directly consumed roughly 17.4 billion gallons of water—a figure that could potentially double or quadruple by 2028 as demand grows. By comparison, American golf courses use more than 500 billion gallons every year for irrigation, often in arid regions where this usage is widely criticized as wasteful. Even if data-center water demand were to grow exponentially, it would take decades to reach the scale of golf-course irrigation.

On the energy side, data centers are indeed taking a larger share of electricity demand. According to the International Energy Agency's latest analysis, they consumed approximately 415 TWh in 2024—about 1.5% of global electricity consumption. This is projected to more than double to around 945 TWh by 2030 (just under 3% of global electricity). However, even this rapid growth accounts for less than 10% (roughly 8%) of the total expected increase in worldwide

electricity demand through 2030. Data centers are therefore not the main driver of the much larger rise in overall energy use.

If we let engineers in Australia, Canada, or Argentina be free to innovate, we will surely see fantastic developments.

You might also have heard about the possibility that ChatGPT might decide to kill us all. Nobody can predict the future, but you are surely more likely to be killed by cancer than by a rogue AI. And AI might help you with your cancer.

We always have a choice. Nations can try to regulate AI out of existence. We can set up new government bodies to prevent the application of AI. This will surely dampen the productivity gains and marginalize some nations economically.

The European Union showed it could be done. By some reports, Europeans make more money by fining American software companies than by building their own innovation enterprises. Countries like Canada have economies dominated by finance, mining and oil (with a side of Shopify).

If you are already well off, stopping innovation sounds good—but not if you are trying to get a start.

AI is likely to help young people who need it so much. They, more than any other group, will find it easier to occupy the new jobs, start the new businesses.

If you are a politician and you want to lose the vote of young people: make it difficult to use AI. It will crater your credibility.

It is time to renew our prosperity. It is time to create new exciting jobs.

References

Wynne, M. A., & Derr, L. (2025, June 24). Advances in AI will boost productivity, living standards over time. *Economic Research*. Federal Reserve Bank of Dallas.
https://www.dallasfed.org/research/economics/2025/0624

Fraser Institute. (2025, December 16). Canada's recent economic growth performance has been awful. *Fraser Institute Commentary*. https://www.fraserinstitute.org/commentary/canadas-recent-economic-growth-performance-has-been-awful

DemandSage. (2026, January 9). AI in education statistics 2026. *DemandSage Research*. https://www.demandsage.com/ai-in-education-statistics/

MIT Technology Review. (2026, January 21). Rethinking AI's future in an augmented workplace. *MIT Technology Review*. https://www.technologyreview.com/

Davis, J. H. (2025). *Coming into view: How AI and other megatrends will shape your investments*. Wiley. https://www.wiley.com/en-us/Coming+into+View-p-9781394251234

Choi, J. H., & Xie, C. (2025, June 26). AI is reshaping accounting jobs by doing the boring stuff. *Stanford Graduate School of Business Insights*. https://www.gsb.stanford.edu/insights/ai-reshaping-accounting-jobs-doing-boring-stuff

International Energy Agency. (2025). Energy demand from AI. In *Energy and AI*. International Energy Agency. https://www.iea.org/reports/energy-and-ai/energy-demand-from-ai

University of Colorado Anschutz Medical Campus. (2025, May 19). Real talk about AI and advancing cancer treatments. *CU Anschutz News*. https://news.cuanschutz.edu/

International Energy Agency. (2025). *Global energy review 2025*. International Energy Agency. https://www.iea.org/reports/global-energy-review-2025

Originally published · February 15, 2026

Let us bury the linear model of innovation

There is an extremely naive model of science and innovation called the linear model: The model postulated that innovation starts with basic research, is followed by applied research and development, and ends with production and diffusion. According to Godin (2006), the model has been falsely attributed to Bush and its dominance is derived rather from its remarkable simplicity. It aligns with a technocratic worldview where bureaucrats get to steer science and innovation.

Yet it has been long documented that innovations arise from practical experience and are driven by user demand. My favorite example is the industrial revolution which was driven by textiles, which, in turn, served to provide comfortable underwear for women. We invented modern-day keyboards after musicians perfected the keyboard for musical instruments.

Kline and Rosenberg are eloquent on this topic:

"Even more important, from the viewpoint of understanding innovation, is the recognition that when the science is inadequate, or even totally lacking, we still can, do, and often have created important innovations, and innumerable smaller, but cumulatively important evolutionary changes. Recently, a member of the National Academy of Engineering, highly versed in dynamics and control, attempted to analyze the stability of an ordinary bicycle with a rider—and failed. No acceptable analysis is known. But this lack of theory did not prevent the invention of the bicycle a century ago, nor did it prevent a multitude of beta-phase improvements in bicycle design that have cumulatively created a reliable, relatively inexpensive, and useful machine. Nor does the absence of understanding of the theory of stability

prevent a 5-year-old child from mounting a bicycle and with a few tries learning to stabilize the human-machine system. Had the idea been true that science is the initiating step in innovation, we would never have invented the bicycle."

You do not need a long catalog after that. Steam engines came out of flooded mines, not lecture halls. Watt improved them under commercial pressure on real machines; a complete theory of heat was optional. The Wright brothers were bicycle mechanics. They flew before anyone could write down a satisfactory theory of a bicycle with a rider—let alone an airplane. Theory followed practice, when it followed at all.

The pattern is reliable enough to be boring: a concrete problem, an imperfect try, users and iteration, and only later the clean story that begins with "basic research." After the fact, institutions rewrite the timeline so that the blackboard appears first. It rarely did.

It is high time we bury the linear model of innovation. Researchers are not in the driving seat of technological innovation, nor are government bureaucrats.

Technology is culture

So where does it actually come from?

We are experiencing one of the most significant technological breakthroughs of the last few decades. Call it what you will: AI, generative AI, large language models…

Technology is culture. Technological progress does not follow a path from the blackboard of a middle-aged MIT professor to your desk, via a corporation.

So what is the cultural background? Of course, there is hacker culture and the way hackers won a culture war in the 1980s by becoming cool enough to have a seat at the table.

But closer to us… I believe there are two main roots. The first is gaming. Gamers wanted photorealistic, high-performance games. They built powerful machines capable of solving linear algebra problems at very high speeds.

Powerful computing alone, however, does you no good if you want to build an AI. That's where web culture came in. Everything was networked, published, republished. Web nerds helped build the greatest library the world had ever seen.

These two cultures came together to generate the current revolution.

If you like my model, I submit that it has a few interesting consequences. The most immediate one is that if you want to understand how and where technological progress happens, you have to look at cultural drivers—not at what professors at MIT are publishing.

References

Godin, B. (2006). The linear model of innovation: The historical construction of an analytical framework. *Science, Technology, & Human Values*, 31(1), 31–60. https://doi.org/10.1177/0162243905283100

Kline, S. J., & Rosenberg, N. (1986). An overview of innovation. In R. Landau & N. Rosenberg (Eds.), *The positive sum strategy: Harnessing technology for economic growth* (pp. 275–306). National Academy Press.

Originally published in two pieces · June 12, 2025 and January 1, 2026

Models and science

The ancient Greeks crafted extraordinary models that continue to resonate. For instance, Ptolemy's geocentric model, with Earth at the core and planets tracing intricate epicycles, elegantly accounted for celestial motion, much like the ancient ambition behind the Tower of Babel sought to bridge humanity and the heavens through a grand, unifying structure. Ptolemy's framework dominated astronomy for over a millennium until Copernicus upended it with a heliocentric vision.

Modern science diverges from these ancient endeavors—whether the Greeks' celestial models or the Babel builders' monumental aspirations—through its insistence on testing ideas against observable reality. The theory of plate tectonics, once dismissed by the scientific establishment, won acceptance not through persuasive arguments but through compelling evidence: seafloor spreading, earthquake distributions, and paleomagnetic records. The Royal Society's motto, Nullius in verba ("Take nobody's word for it"), encapsulates this ethos—no authority, however revered, escapes scrutiny. This defines science: a steadfast commitment to an external, tangible reality that exists beyond human invention.

Richard Feynman captured this principle succinctly: "It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong." Models, no matter how elegant, are mere human constructs, subordinate to the unyielding truth of reality. Science demands humility—the rejection of hubris, whether in the form of a towering edifice or a cherished theory, and an embrace of relentless questioning, empirical rigor, and the courage to discard flawed ideas when evidence dictates.

Originally published · May 23, 2025

As smart as a PhD

There is sometimes confusion about what a PhD is.

The main signal that you should derive from the fact that someone has a PhD is that they are well suited to the university campus environment.

Maybe people who complete a PhD are especially thorough and finish their work to perfection? In computer science, academic projects have a reputation for being of relative quality. Stonebraker, a famous computer scientist, attributes part of his success to his dedication to finishing up the work:

> The smartest things we ever did, was to then put in the effort to make it really work. (Stonebraker, 2014)

Did you know that a lot of people have a PhD?

Over 3% of the population in a country like Switzerland has a PhD. Hence, in a city of 1 million people, you may have 30,000 people with a PhD. In Germany, that would be 15,000 people.

Do PhDs lead to great jobs and higher incomes? Not really. If you do get your PhD and secure a good job and keep it for a long time (so no early retirement), then you can do a bit better than the *average person* who stopped with a professional degree. But even that is misleading because we don't know how the person smart enough to outcompete 100 other PhDs for a prestigious job would have done had they not gone for the PhD.

You know, Joe, who has a PhD and has become a full professor at Prestigious University… well… it is likely that Joe has been working weekends and nights for years. Joe is excessively well connected. Joe was always smarter than anyone else in his classes. Joe can sit

down and write a great 20-page scientific essay without ChatGPT and without much effort. Joe can navigate politics better than most.

Sometimes young people think… well, it is terrible to be job hunting at 22 without experience. Right, but try job hunting at 30 with a PhD and no actual job experience. It is worse! Much worse!

Given these facts, do you really want to be 'as smart as a PhD'?

> "In the short run, pursuing a PhD entails substantial opportunity costs. Early-career earnings for PhD graduates are significantly lower than those of individuals with master's or professional degrees. (…) Over the lifecycle, earnings do eventually recover but only under specific conditions. The most favorable long-run outcomes are concentrated among those who secure academic employment and remain in full-time work late into life. (…) However, the structure of this system increasingly resembles a tournament: the payoff remains high for those who reach the top, but the odds of doing so have declined. Our analysis documents that the economic outcomes of recent PhD graduates have worsened over time. The bottom of the earnings distribution has grown more populated, and early-career returns have declined even as aggregate statistics appear stable due to rising returns among older cohorts." (Benjamin et al., 2025)

References

Stonebraker, M. (2014). How to build a really good database system. In *CIDR 2014, Conference on Innovative Data Systems Research*. https://www.cidrdb.org/cidr2014/papers/cidr14_paper1.pdf

Benjamin, D. J., et al. (2025). The economics of PhDs. *NBER Working Paper* No. 33456. https://doi.org/10.3386/w33456

Originally published · October 30, 2025

Sovereign

The keyword in politics these days is ‘sovereign’.

What few will admit is that it is effectively the adoption of the American strategy: Make America Great Again. In other words, reindustrialization of key sectors of the economy. The UK used to be a computing champion. Our chip designs (ARM) originated from the UK. Canada had BlackBerry, everyone was using Canadian phones.

Like Canada, many countries have progressively slid into financialization. Huge banks and bank-related businesses, surrounded by emptied factories.

Part of it was the doing of economists who promoted globalization. We are going to make our best CPUs in Taiwan, because they have a comparative advantage (whatever that means).

Another part is the rise of the managerial class, or our version of the technocracy: the summum of the status game is to make PowerPoint presentations in a nice office. Everyone has 2 or 3 university degrees. And if you don’t have many degrees, what is wrong with you?

I think that this is breaking apart for a few reasons.

One of them is Trump. And I don’t mean bombastic statements, bad hair color or overly long ties… But rather the realization that globalization might leave your people economically better off for a time because they have cheap stuff… But it also leaves your people with few skills. You can fund a robotics factory near Montreal, and you’ll find 2000 people with robotics PhDs, but nobody actually knows how to build robots. The analogy is thus: salary is not everything (to the great chagrin of economists). I have left a much better paying job. My previous job meant that I had to sit in an office and do little concrete. I would have had a great and early retirement… but I would never have developed my skills nor would I have built anything. And

that's what we did at the country level. Great total compensation, but a dead-end skill-wise.

Another factor, I believe, is the COVID era (2020-2023) and its final outcome: empty offices, closed coffee shops. What happened at work was illegible. Lots of people in offices. Certainly, something important was happening. Entire businesses and government organizations have now migrated partially or entirely to a pajama party of some kind. Netflix in the middle of the workday is no longer a dream, but a reality.

Another element is educational misalignment. A country like Canada has the most schooled population in history. You cannot throw a rock without hitting someone with a PhD. Meanwhile, we are not making robots or microchips. You can't even pay with your phone in the Montreal subway. It is a project for another decade, maybe. We are using a push strategy: push more people with degrees into the economy and you are going to get a fancier economy. Won't work.

Finally, the AI breakthrough of 2022 is the final nail in the proverbial coffin. My country (Canada) claimed for decades that we were the AI powerhouse. All these PDFs online can't lie, can they? Canada basically invented modern AI, didn't it? We did. On paper. On paper we did a great many things. In practice? Few know how to build anything.

In a country like Canada, the population has not yet caught up. They blame the orange man for whatever trouble they see. And the politicians promise to do what they must: shower money to build tech sovereignty. It won't work. They will try again. It won't work again. The cycle makes things worse because it sustains a managerial class that is great at politics but terrible at building.

Meanwhile, you can't escape preference falsification. People will use ChatGPT, Claude, Grok, Gemini. And if Elon produces robots, they'll want them.

Trump will leave office in two years… Canada and the UK will still be flat-lined economically. The USA will still surge ahead.

Countries like Canada and the UK will have to realize that it is industry and know-how first. Build stuff and the wealth will follow. Stop the virtue signaling. Stop the credentialism. Build.

Originally published · June 9, 2026

Foolish enough to leave important tasks to a mere human brain?

When I wrote this in 2015, I guessed that within twenty years we would have machines surpassing the human brain in every way that matters. We are halfway to that horizon. Large language models already beat most of us at drafting text, translating languages, and answering trivia; they still stumble on judgment, taste, and finishing hard research problems. Flight took decades from the Wright brothers to routine ocean crossings; machine cognition may follow a similar curve—sudden demonstrations, then a long, uneven diffusion. The biological-supremacy myth is weaker than it was, but not yet gone.

To the ancient Greeks, the male reproductive organ was mysterious. They had this organ that can expand suddenly, then provide the seed of life itself. Today, much of biology remains uncatalogued and mysterious, but the male reproductive organ is now fairly boring. We know that it can be cut (by angry wives) and sewed back in place, apparently with no loss of function. As for providing the seed of life, artificial insemination is routine both in animals (e.g., cows) and human beings. In fact, by techniques such a cloning, we can create animals, and probably even human beings, with no male ever involved.

If we go back barely more than a century, flight was mysterious. Birds looked slightly magical. Then a couple of bicycle repairmen, who dropped out of high school, built the first airplane. Today, I would not think twice about embarking in a plane, with hundred of other people, and fly over the ocean in a few hours… something no bird could ever do.

This is a recurring phenomenon: we view something as magical, and then it becomes a boring mechanism that students learn in textbooks. I call it the biological-supremacy myth: we tend to overestimate the complexity of anything biology can do… until we find a way to do it ourselves.

Though there is still much we do not know about even the simplest functions of our body, the grand mystery remains our brain. And just like before, people fall prey to the biological-supremacy myth. Our brains are viewed as mythical organs that are orders of magnitude more complex than anything human beings could create in this century or the next.

We spend a great deal of time studying the brain, benchmarking the brain, in almost obsessive ways. Our kids spend two decades being tested, retested, trained, retrained… often for the sole purpose of determining the value of the brain. Can't learn calculus very well? Your brain must not be very smart. Can't learn the names of the state capitals? Your brain must be slightly rotten.

In the last few years, troubles have arisen for those who benchmark the brain. I can go to Google and ask, in spoken English, for the names of the state capitals, and it will give them to me, faster than any human being could. If I ask Google "what is the derivative of sin x", not only does it know the answer, it can also point to complete derivation of the result. To make matters worse, the same tricks work anytime, anywhere, not just when I am at the library or at my desk. It works everywhere I have a smartphone, which is everywhere I might need calculus, for all practical purposes.

What is fascinating is that as we take down the brain from its pedestal, step by step, people remain eager to dismiss everything human-made as massively inferior:

- "Sure, my phone can translate this Danish writing on the wall for me, but it got the second sentence completely wrong. Where's your fantastic AI now?"
- "Sure, I can go to any computer and ask Google, in spoken English, where Moldova is, and it will tell me better than a human being could… But when I ask it when my favorite rock band was playing

again, it cannot figure out what my favorite rock band was. Ah! It is a joke!"

A general objection regarding the brain is that there is so much we do not know. As far as I can tell, we do not know how the brain transforms sounds into words, and words into ideas. We know which regions of the brains are activated, but we do not fully understand how even individual neurons work.

People assume that to surpass nature, we need to fully understand it and to further fully reproduce it. The Wright brothers would have been quite incapable of modeling bird flight, let alone reproduce it. And a Boeing looks like no bird I know… and that's a good thing… I would hate to travel on top of a giant mechanical bird.

Any programmer will tell you that it can be orders of magnitude easier to reprogram something from scratch, rather than start from spaghetti code that was somehow made to work. We sometimes have a hard time matching nature, not because nature was so incredibly brilliant… but rather because, as an engineer, nature is a terrible hack: no documentation whatsoever, and an "if it works, it is good enough" attitude.

This same objection, "there is so much we do not know", is used everywhere by pessimists. Academics are especially prone to fall back on this objection, because they like to understand… But, of course, all the time, we develop algorithms and medical therapies that work, without understanding everything about the problem. That's the beautiful thing about the world we live in: we can act upon it in an efficient manner without understanding all of it.

Our puny brains may never understand themselves, but that does make our brain wonderful and mysterious… it is more likely the case that our brains are a hack that works well enough, but that is far from the best way to achieve intelligence.

Another mistake people make is to assume that evolution is an optimization process that optimizes for what we care about as human beings. For centuries, people thought that if we were meant to fly, we would have wings. Evolution did not give us wings, not as a sign that

we couldn't fly… but simply because there was no evolutionary path leading to monkeys with wings.

Similarly, there is no reason to believe that evolution optimized human intelligence. It seems that other human species had larger brains. Our ancestors had larger brains. Several monkeys have photographic memory, much better strength/mass ratios and better reflexes. The human body is nothing special. We are not the strongest, fastest and smartest species to ever roam the Earth. It is likely that we came to dominate the animal kingdom because, as a species, we have a good mix of skills, and as long as we stay in a group, we can take down any other animal because we are expert at social coordination among mammals.

Yes, it is true that evolution benefited from a lot of time… But that's like asking a programmer to tweak a piece of code randomly until it works. If you give it enough time, the result will work. It might even look good from the outside. But, inside, you have a big pile of spaghetti code. It is patiently tuned code, but still far from optimality from our point of view.

The Wright brothers were initially mocked. This reassured the skeptics that believed that mechanical flight was a heresy. But, soon after, airplanes flew in the first world war.

In 20 years, we will have machines that surpass the human brain in every way that matters to us. It will look nothing like a human brain… probably more like a Google data warehouse at first… And then we will be stuck with the realization that, from our reproductive organs all the way to our brains, we are nothing special.

Many people refuse to believe that we will ever have machines that are better than us in every way. And they are right to be scared because once you invent a machine that is smarter than you are, you have no choice: you have to put it in charge.

Human beings know that they are somehow irrelevant in the grand scheme of things. I write these words using a brain that consumes maybe 20 to 30 Watts, with the bulk of my neurons actually invested in controlling my body, not thinking abstractly. In a few decades, it will be trivial to outsmart me. And then I will be back to being an old,

boring monkey… no longer a representative of the smartest species on Earth.

Of course, just because we do not need the male organ to procreate does not mean that people stop having sex. The birds did not stop flying when we invented the airplane. Television did not mean the end of radio. The Internet does not mean the end of the paper. Hopefully, my species will make use of its brains for many decades, many centuries… but soon enough, it will seem foolish to leave important decisions and tasks to a mere human brain.

Originally published · October 5, 2015

The Craft of Software

These seven short essays close the book. They are not a third equal act beside the two main parts, but a brief application of the same temperament to software itself: whether performance still matters when hardware is fast; why optimizing “hotspots” is not enough; why large user communities do not automatically produce great engineering teams; and what habits separate durable systems from fragile ones.

Does software performance still matter?

Someone once asked me about the real-world relevance of software performance:

> I'm quite interested in your work on improving algorithm performance using techniques related to computer architecture. However, I think that this may only be of interest to academia. Do you think that there are job opportunities related to this profile, which is very specialized?

To paraphrase the question, computers and software are fast enough. We may need people to implement new ideas, but performance is not important. And more critically, if you want to be gainfully employed, you do not need to worry about software performance.

To assess this question, we should first have a common understanding of what software performance is. Software performance is not about how quickly you can crunch numbers. It is how you manage memory, disks, networks, cores… it is also about architecture. It is not about rewriting your code in machine code: you can write fast applications in JavaScript and slow ones in C++. Software performance is related to algorithmic design, but distinct in one important way: you need to take into account your architecture. Many algorithms that look good on paper do really poorly in practice. And algorithms that appear naive and limited can sometimes be the best possible choice for performance. In some sense, being able to manage software performance requires you to have a good understanding of how computer hardware, operating systems, and runtime libraries work.

So, should we care about software performance in 2017?

Here is my short answer. There are two basic ways in which we can assess you as a programmer. Is your software correct? Is your software efficient? There are certainly other ways a programmer can bring value: some exploit their knowledge of some business domains, others will design marvelous user interfaces. However, when it comes down to hardcore programming, being correct and being efficient are the two main attributes.

Consider the great programmers. They are all good at producing software that is both correct and efficient. In fact, it is basically the definition of a great programmer. Programming is only challenging when you must be both correct and efficient. If you are allowed to sacrifice one or the other, you can trivialize most tasks.

In job ads, you probably won't see many requests for programmers who write efficient code, nor are you going to see many requests for programmers who write correct code. But then, you do not see many ads for doctors who cure people, nor do you see many ads for lawyers who avoid expensive lawsuits. Producing efficient code that takes into account the system's architecture is generally part of your job as a programmer.

Some of my thoughts in detail:

- **Software performance only matters for a very small fraction of all source code. But what matters is the absolute value of this code, not its relative size.** Software performance is likely to be irrelevant if you have few users and little data. The more important the software is, the more important its performance can become.

Given that over 90% of all software we write is rarely if ever used for real work, it is a safe bet to say that software performance is often irrelevant, but that's only because, in these cases, the software brings little value.

Let us make the statement precise: **Most performance or memory optimizations are useless.**

That's not a myth, it is actually true.

The bulk of the software that gets written is not performance sensitive or worth the effort. Pareto's law (Newman, 2005) would tell you that 20% of the code accounts for 80% of the running time, but I think it is much worse than this. I think that 1% of the code accounts for 99% of the running time... The truth is maybe even more extreme.

So a tiny fraction of all code will ever matter for performance, and only a small fraction of it brings business value.

But what matters, if you are an employee, is how much value your optimized code brings to the business, not what fraction of the code you touch.

- **We can quantify the value of software performance and it is quite high.** If I go on Apple's website and I shop for a new MacBook Pro. The basic one is worth $1,800. If I want a processor with a 10% faster clock speed, it is going to cost me $2,100, or 15% more. An extra 10% in the clock speed does not make the machine nearly 10% faster. Let us say that it is maybe 5% faster. So to get a computer that runs 5% faster (if that) some people are willing to pay 15% more. I could do the same analysis with smartphones.

If constant factors related to performance did not matter, then a computer running at twice the speed would be worth the same. In practice, a computer running at twice the speed is worth multiple times the money.

With cloud computing, companies are now often billed for the resources (memory, compute time) that they use. Conveniently, this allows them to measure (in dollars) the benefits of a given optimization. We can find the stories of small companies that save hundreds of thousands of dollars with some elementary optimization.

We could also look at web browsers. For a long time, Microsoft had the lead with Internet Explorer. In many key markets, Google Chrome now dominates. There are many reasons for people to prefer Google Chrome, but speed is a key component. To test out my theory, I searched Google for guides to help me choose between Chrome and Internet Explorer, and the first recommendation I found was this:

> Chrome is best for speed arguably, a web browser's most crucial feature is its ability to quickly load up web pages. We put both Chrome and Internet Explorer 11 through a series of benchmark tests using Sunspider, Octave and HTML 5 test. In every event, Google's Chrome was the clear winner.

So yes, performance is worth a lot to some users.

- **Adding more hardware does not magically make performance issues disappear. It requires engineering to use more hardware.** People object that we can always throw more machines, more cores at the problem if it is slow. However, even when Amdahl's law (Amdahl, 1967) does not limit you, you still have to contend with the fact that it can be hard to scale up your software to run well on many machines. Throwing more hardware at a problem is just a particular way to boost software performance. It is not necessarily an inexpensive approach.

It should be said that nobody ever gets an asymptotically large number of processors in the real world. Moreover, when you do get many processors, coordination issues can make it difficult (even in principle) to use a very large number of processors on the same problem.

What about our practice? What the past decades have taught us is that parallelizing problems is hard work. You end up with more complex code and non-trivial overhead. Testing and debugging get a lot more difficult. With many problems, you are lucky if you manage to double the performance with three cores. And if you want to double the performance again, you might need sixteen cores.

This means that doubling the performance of your single-threaded code can be highly valuable. In other words, tuning hot code can be worth a lot… And adding more hardware does not make the performance problems go away magically, using this hardware requires extra work.

- **We use higher-level programming languages, but an incredible amount of engineering is invested in recovering the traded-away performance.** Today's most popular programming language is JavaScript, a relatively slow programming language. Isn't that a sign that performance is irrelevant? The performance of JavaScript was multiplied over the years through vast engineering investments. Moreover, we are moving forward with high-performance web programming techniques like Web Assembly. If performance did not matter, these initiatives would fall flat.

It is true that, over time, people migrate to high-level languages. It is a good thing. These languages often trade performance for convenience, safety or simplicity.

But the performance of JavaScript in the browser has been improved by two orders of magnitude in the last fifteen years. By some estimates, JavaScript is only about ten times slower than C++.

I would argue that a strong component in the popularity of JavaScript is precisely its good performance. If JavaScript was still 1000 times slower than C++ at most tasks, it would not have the wide adoption we find today.

Last year, a colleague faced a performance issue where simulations would run forever. When I asked what the software was written in... she admitted with shame that it was written in Python. Maybe to her surprise, I was not at all dismissive. I'd be depressed if, in 20 years, most of us were still programming in C, C++, and Java.

One of the things you can buy with better performance is more productivity.

- **Computers are asked to do more with less, and there is a never-ending demand for better performance.** Software performance has been regularly dismissed as irrelevant. That's understandable under Moore's law (Moore, 1965): processors get faster, we get faster disks... who cares if the software is slow? It will soon get faster. Let us focus on writing nice code with nice algorithms, and we can ignore the rest.

It is true that if you manage to run Windows 3.1 on a recently purchased PC, it will be ridiculously fast. In fact, I bet you could run Windows 3.1 in your browser and make it fast.

It is true that some of the hardware progress reduces the pressure to produce very fast code… To beat the best chess players in the 1990s, one probably needed the equivalent of hand-tuned assembly code whereas I am sure I can write a good chess player in sloppy JavaScript and get good enough performance to beat most human players, if not the masters.

But computers are asked to do more with less. It was very impressive in 1990 to write a Chess program that could beat the best Chess players… but it would simply not be a great business to get into today. You'd need to write a program that plays Go, and it is a lot harder.

Sure, smartphone hardware gets faster all the time… but there is pressure to run the same software on smaller and cheaper machines (such as watches). The mouse and the keyboard will look quaint in a few decades, having been replaced by more expensive interfaces (speech, augmented reality…).

And yes, soon, we will need devices the size of a smartwatch to be capable of autonomous advanced artificial intelligence. Think your sloppy unoptimized code will cut it?

- **We do want our processors to be idle most of the time. The fact that they are is not an indication that we could be sloppier.** Aren't most of our processors idle most of the time?

True, but it is like saying that it is silly to own a fast car because it is going to spend most of its time parked.

We have enormous overcapacity in computing, by design. Doing otherwise would be like going to a supermarket where all of the lines for the cashiers are always packed full, all day long.

We have all experienced what it is to use a laptop that has its CPU running at 100%. The laptop becomes sluggish, unresponsive. All your requests are queued. It is unpleasant.

Servers that are running at full capacity have to drop requests or make you wait. We hate it.

A mobile phone stressed to its full capacity becomes hot and burns through its battery in no time.

So we want our processors to be cold and to remain cold. Reducing their usage is good, even when they were already running cold most of the time. Fast code gets back to the users faster and burns less energy.

I imagine that the future will be made of dark silicon. We will have lots of computing power, lots of circuits, but most of them will be unpowered. I would not be surprised if we were to soon start rating software based on how much power it uses.

References

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *AFIPS Conference Proceedings* (Vol. 30, pp. 483–485).

Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, 38(8), 114–117.

Newman, M. E. J. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46(5), 323–351. https://doi.org/10.1080/00107510500052444

Originally published · March 20, 2017

Hotspot performance engineering fails

Developers often believe that software performance follows a Pareto distribution (Newman, 2005): 80% of the running time is spent in 20% of the code.

Using this model, you can write most of your code without any care for performance and focus on the narrow pieces of code that are performance sensitive. Engineers like Casey Muratori have rightly criticized this model. You can read Muratori's excellent piece, *Performance Excuses Debunked*.

It is definitely true that not all of your code requires attention. For example, it is possible that 99% of the time, your code processes correct data. The code that handles errors could be quite slow and it would not impact most of your users.

But the hotspot predicts something more precise: you should be able to just keep all of your code, identify the specific bottlenecks, optimize these bottlenecks, and then get great performance all around. Muratori relies on empirical evidence to falsify the model: many companies embarked on large rewrites of their codebase to optimize it. Why bother with such expenses if they could simply identify the bottlenecks and fix those?

We have tools today called profilers that can tell you roughly where your software spends its time. And if you apply such a tool to your software, you may indeed find massive bottlenecks. It may sometimes work wonders. For example, there was a video game (GTA Online) that was loading a JSON file. Simply optimizing this one bottleneck solved a massive performance issue. It did not make the game more performant, but it made it start much faster. So bottlenecks do exist.

We should hunt them down and optimize them. But that's not what the hotspot model predicts: it predicts that it is all you need to do to get performance. Hit a few bottlenecks and you are done.

Sadly, it does not work. To illustrate, let us look at how the very popular SQLite project got to be 50% faster:

> The latest SQLite 3.8.7 alpha version is 50% faster than the 3.7.17 release from 16 months ago. That is to say, it does 50% more work using the same number of CPU cycles. This is 50% faster at the low-level grunt work of moving bits on and off disk and search b-trees. We have achieved this by incorporating hundreds of micro-optimizations. Each micro-optimization might improve the performance by as little as 0.05%. If we get one that improves performance by 0.25%, that is considered a huge win. Each of these optimizations is unmeasurable on a real-world system but if you do enough of them, they add up.

Why did they have to add up hundreds of micro-optimizations instead of simply finding the one function that uses 80% of the running time?

Let us run down through a few reasons:

- **As you optimize, the hotspots multiply**. Going back to the example of GTA Online, it is easy to find that the program spends 10 seconds loading a 10 MB JSON file. However, the next steps are going to be more difficult. You will find that finding the bottlenecks becomes difficult: we are subject to a Heisenberg principle: measuring big effects is easy, measuring small ones becomes impossible because the action of measuring interacts with the software execution. But even if you can find the bottlenecks, they become more numerous with each iteration. Eventually, much of your code needs to be considered.
- **Overall architecture trumps everything**. If you first build a bus, you cannot then turn it into a sports car with a few changes. A few years ago, a company came to me and offered me a ton of money if I could make their database engine faster. They had money, but

> their software was much too slow. At first I was excited by the project, but I started reading their code and doing benchmarks, and then I realized that the entire architecture was wrong. They insisted that they knew where the hotspots were and that they just needed the expertise to optimize these few components. They told me that their code was spending 80% of its time in maybe 100 lines. And that is what profilers said. It is true, formally speaking, that if you could have made these 100 lines of code twice as fast, the code would have run twice as fast… but these lines of code were pulling data from memory and software cannot beat Physics. The key point is that if your software does not have a good overall architecture, if it is not well organized for performance, you may have no choice but to rewrite it at least in part, to re-architect it. This means that even though there might be a very clear hotspot involving few lines of code, fixing it might require changing almost all your code.

The effort needed to optimize code grows exponentially. In other words, to multiply the performance by N, you need 2^N optimizations. The more you optimize, the more code you must consider. It is relatively easy to double the performance of an unoptimized piece of code, but much harder to multiply it by 10. In effect, we have a long tail effect: though there are clearly easy wins, much of the work lies in the 'tail'. In effect, you end up winning by continually adding small optimizations.

References

Muratori, C. (2021). Performance excuses debunked. *Computer Enhance*. https://www.computerenhance.com/p/enjoy-your-compiler

Lois, F. (n.d.). Patterns for high-performance C#. Conference presentation.

Tinn, A. (2021, March 7). Hacker reduces GTA Online load times by over 70 percent. *Ars Technica*.

https://arstechnica.com/gaming/2021/03/hacker-reduces-gta-online-load-times-by-over-70-percent/

SQLite Development Team. (2014). SQLite 3.8.7 release notes. *SQLite*. https://www.sqlite.org/releaselog/3_8_7.html

Anderson, C. (2004). The long tail. *Wired*, 12(10). https://www.wired.com/2004/10/tail/

Newman, M. E. J. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46(5), 323–351. https://doi.org/10.1080/00107510500052444

Originally published · April 27, 2023

Metcalfe's law against Brooks' law

Guido van Rossum (2025), Python's creator, recently said: "We have a huge community, but relatively few people, relatively speaking, are contributing meaningfully."

This highlights a paradox.

Software thrives on the network effect, or Metcalfe's Law (Metcalfe, 2013), where a system's value scales with the square of its users. Linux excels because its vast user base fuels adoption, documentation, and compatibility everywhere.

But larger teams don't build better software—often the reverse. Brooks' Law (Brooks, 1995) shows that adding people increases communication overhead, slowing progress. The Pareto Principle (Newman, 2005) also applies: a small minority drives most meaningful contributions. Great software often stems from a single visionary or a small, cohesive team, not a crowd.

The network effect applies primarily to users, not necessarily to creators.

References

Metcalfe, R. M. (2013). Metcalfe's law after 40 years of ethernet. *Computer*, 46(12), 26–31. https://doi.org/10.1109/MC.2013.374

Brooks, F. P. (1995). *The mythical man-month: Essays on software engineering* (Anniversary ed.). Addison-Wesley.

van Rossum, G. (2025, June 12). Is "worse is better" still better? [Conference presentation]. Python Language Summit 2025. https://pyfound.blogspot.com/2025/06/python-language-summit-2025-lightning-talks.html

Newman, M. E. J. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46(5), 323–351. https://doi.org/10.1080/00107510500052444

Originally published · June 15, 2025

Don’t be so eager to rewrite your code

I used to always want to rewrite my code. Maybe even use another programming language. “If only I could rewrite my code, it would be so much better now.”

If you maintain software projects, you see it all the time. Someone new comes along and they want to start rewriting everything. They always have subjective arguments: it is going to be more maintainable or safer or just more elegant.

If your code is battle tested… then the correct instinct is to be conservative and keep your current code. Sometimes you need to rewrite your code: you made a mistake or must change your architecture. But most times, the old code is fine and investing time in updating your current code is better than starting anew.

The great intellectual Robin Hanson argues that software ages. One of his arguments is that software engineers say that it does. That’s what engineers feel but whether it is true is another matter.

“Before Borland’s new spreadsheet for Windows shipped, Philippe Kahn, the colorful founder of Borland, was quoted a lot in the press bragging about how Quattro Pro would be much better than Microsoft Excel, because it was written from scratch. All new source code! As if source code rusted. The idea that new code is better than old is patently absurd. Old code has been used. It has been tested. Lots of bugs have been found, and they’ve been fixed. There’s nothing wrong with it. It doesn’t acquire bugs just by sitting around on your hard drive. Au contraire, baby! Is software supposed to be like an old Dodge Dart, that rusts just sitting in the garage? Is software like a teddy bear

that’s kind of gross if it’s not made out of all new material?” (Joel Spolsky)

References

Spolsky, J. (2000, April 6). Things you should never do, Part I. *Joel on Software*. https://www.joelonsoftware.com/2000/04/06/things-you-should-never-do-part-i/

Originally published · December 28, 2025

Fast software is a discipline, not a purpose

When people train, they usually don't try to actually run faster or lift heavier weights. As a relatively healthy computer science professor, how fast I run or how much I can lift is of no practical relevance. However, whether I can walk the stairs without falling apart is a metric.

I am not an actor or a model. Who cares how much I weigh? I care: it is a metric.

I could probably work in a dirty office without ill effect, but I just choose not to.

So when I see inefficient code, I cringe. I am being told that it does not matter. Who cares? We have plenty of CPU cycles. I think you should care, it is a matter of discipline.

Yes, only about 1% of all code being written really matters. Most people write code that may as well be thrown out.

But then, I dress cleanly every single day even if I stay at home. And you should too.

I do not care which programming language you use. It could be C, it could be JavaScript. If your code is ten times slower than it should be, I think it shows that you do not care, not really. And it bothers me. It should bother you because it tells us something about your work. It is telling us that you do not care, not really.

Alexander Jay sent me a nice email. He reviewed some tricks he uses to write fast code. It inspired these recommendations in me:

- Avoid unnecessary memory allocations.
- Avoid multiple passes over the data when one would do.
- Avoid unnecessary runtime inferences and branches.

- Avoid unnecessary performance-adverse abstraction.
- Prefer simple value types when they suffice.
- Learn how the data is actually represented in bits, and learn to dance with these bits when you need to.

Alexander asked me, “At what point would you consider the focus on optimization a wasted effort?” My answer: “At what point do you consider being fit and clean a wasted effort?”

There is a reason we don’t tend to hire people who show up to work in dirty t-shirts. It is not that we particularly care about dirty t-shirts, it is that we want people who care about their work.

If you want to show care for your software, you first have to make it clean, correct and fast. If you start caring about bugs and inefficiencies, you will write good software. It is that simple.

Originally published · November 16, 2017

Don’t argue: build

Prescientific and preindustrial thought tied truth to authority and tradition. “We’ve always done it this way.” “The king decrees it.” “We know this is how it’s done.”

The scientific and industrial revolutions shattered this mindset. Suddenly, outsiders could challenge entrenched norms. Two brothers in a rundown workshop could build empires to rival the wealthiest lords. Ordinary people could question the highest authorities.

What made this possible? A belief in objective truth. Reality exists independently of our perceptions. For Christians, this meant God created a discoverable world, and humanity’s role was to uncover its workings and harness them.

How could an individual stand against tradition or authority? Through facts. A new method is cheaper—or it isn’t. It’s faster—or it isn’t. It yields better results—or it doesn’t.

This is the essence of progress: you can improve, and you can prove it.

The British Empire dominated the seas, waged wars, and abolished slavery not because of superior manpower or land, but through better governance, business practices, and science.

Not every question is empirical, but the critical ones often are. To make your case, build it and demonstrate its value. That is the heart of the scientific revolution.

In software, the same habit applies. Whether a design will scale, whether a database can serve a million queries per second, whether a rewrite is worth the risk—these are not questions that yield to seniority or eloquence in a meeting. Prototype the approach, benchmark it against what you have, ship and measure in production. When the numbers disagree with the architecture review, the numbers win. That

is not hostility to debate; it is the discipline that keeps building from becoming performance art.

Originally published · June 16, 2025

Programmer time and the pitfalls of false positives

Programmer time is precious. This realization should shape our approach to software development, focusing our efforts on tasks that genuinely contribute to the improvement of our code and the software ecosystem.

What does matter?

1. Hunting for bugs. I like to add tests, and then even more tests. The time spent building tests should be proportionate to the time spent building the software. Fuzzing is also fantastically useful. I love using sanitizers.
2. Fixing bugs. Bugs disrupt user experience, compromise functionality, and can even introduce security vulnerabilities. Addressing bugs is critical to build trust in the software.
3. Documentation matters. Underdocumented code is mysterious and may trigger unnecessary surprises. Lack of documentation may also harm relationships with users.
4. Adding new features. Innovation and growth in software come from introducing new features. Features should be user visible: 'internal' features are often wasteful.
5. Improving Performance. Performance enhancement is all about making the software run faster, use fewer resources, or handle larger workloads more efficiently. This can significantly impact user satisfaction, particularly in applications where speed is paramount. Improving performance is not just about identifying bottlenecks... it is an ongoing journey. You need a good design and multiple rounds of optimizations. You can often continue to improve the performance for years and years.

However, there are areas where I believe our time is not well spent:

- Patching code to silence false positives from disabled-by-default static analyzers. The level 4 warnings under Visual Studio when compiling C++ code are a good example, but so are the obscure GCC and clang warnings. Static analyzers are tools that can scan code for potential issues without executing the program. However, when these tools are overly strict or misconfigured, they might report numerous false positives; issues that aren't actually problems. Spending time patching code merely to quiet these false alarms is, in my view, wasteful. It diverts attention from more impactful work. This is not to say that static analysis is not beneficial; when used correctly, it can save considerable time and resources. But the effort required to address non-issues can quickly become counterproductive.
- Aimless refactoring is also often wasteful. Renaming classes, moving code around just so that it looks 'nice'. I am not against the occasional cleaning round… but it should not be time consuming. Refactoring for its own sake may become an excuse for not fixing bugs or for not improving the performance. It is easy work, but often not impactful.

Let us take a concrete example. Consider the following function:

```
uint8_t f(uint64_t x) {
 uint8_t y = x % 256;
 return y;
}
```

When compiled with LLVM/clang with the -Wconversion flag, you get the following warning:

warning: implicit conversion loses integer precision: 'uint64_t' (aka 'unsigned long') to 'uint8_t' (aka 'unsigned char') [-Wimplicit-int-conversion]

The warning is bogus, a waste of time. There is no loss of precision (whatever precision means).

Another favorite of mine is this function:

```
uint64_t g(uint64_t x) {
 return -x;
}
```

Negating an unsigned integer has many applications: e.g., if you want to select only the least significant bit, and zero all others, you can use the expression (-x) AND x. Under Visual Studio, if you crank up the warnings, you get

warning C4146: unary minus operator applied to unsigned type, result still unsigned

Another waste of time.

While we strive for perfection in our code, we must also be strategic about where we invest our most precious resource: programmer time. Let us prioritize what truly matters in the grand scheme of software development.

Originally published · January 30, 2025

Afterword

Forty-three essays and two interludes cannot exhaust a career's worth of questions. They can, I hope, make a pattern visible: protect your attention, learn by doing, measure honestly when it matters, and stay open to what you do not yet understand.

The two main parts of this book—and the short software close—are one stance seen from different angles. Productivity is not a hack; optimism is not naïveté; performance is not nostalgia. Each comes from paying attention to what actually works—and acting on it.

The future remains uncertain. Uncertainty cuts both ways. If disciplined thinking and patient engineering beat conventional limits more often than we expect, then the future is more worth building toward than cynicism suggests.

About the Author

Daniel Lemire is a computer science professor and software performance expert. His work on fast algorithms and data structures—including fast float parsing, Roaring bitmaps, and SIMD-accelerated techniques—has been adopted in databases, browsers, and production systems worldwide. He ranks among the top 2% of scientists globally (Stanford/Elsevier, 2025) and is one of GitHub's most followed developers.