Tips For Aspiring Data Scientists

by Manuel Amunategui

How to become a data scientist
How to find your next programming language
How to stay creative in this fast-paced profession
and much, much more

My #1 Piece of Advice for Aspiring Data Scientists

Published on May 21, 2018 on Medium.com

(Source: Lucas Amunategui)

Whenever I am asked how to break into this field, I reply, ‘push it out into the public domain on a regular basis’.

Being a blogger and liking the sound of my voice, I unfortunately murk it up with more stuff, like taking classes on Udacity and Coursera, participating in competitions on Kaggle, and getting a degree in data science. But, at the top of the list, will always be, create, invent and push out regularly!
It may go against the grain within some organizations, so don’t share trade secrets or other people’s IP; but there’s always something within your skillset that others will find interesting.

It’s a miracle technique! It’s isn’t about quantity or quality but about doing it on a regular basis and building that production muscle—it becomes magic! It will take you to unexpected places. Kind of like compounding interest or butter in your coffee—there’s no downside to this approach!

Self-Development

You will learn to express yourself better, write better, and put yourself out

there. You will gain a fuller grasp of the topics discussed—there is nothing like explaining it to others to learn it better yourself.

Great For Finding Jobs And Networking

You will be sending the clear message that you enjoy this stuff and spend time on it. That you are motivated, entrepreneurial, and you know your stuff—what kind of employer or customer could resist that? You will meet a lot of people, not just students but a wide range of professionals as well. You will be invited to speak at conferences and meet ups.

Great For Your Current Job

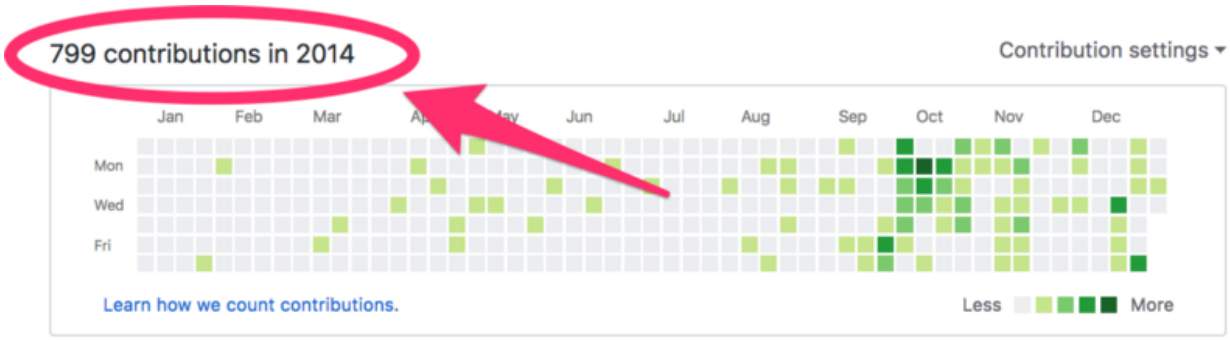
These may be stories for another entry, but I have connected with colleagues within a large health system through my You Tubes, I kid you not—people I would email directly and never get a reply—sometimes a social media knock is louder than an office door knock.

You become Accountable

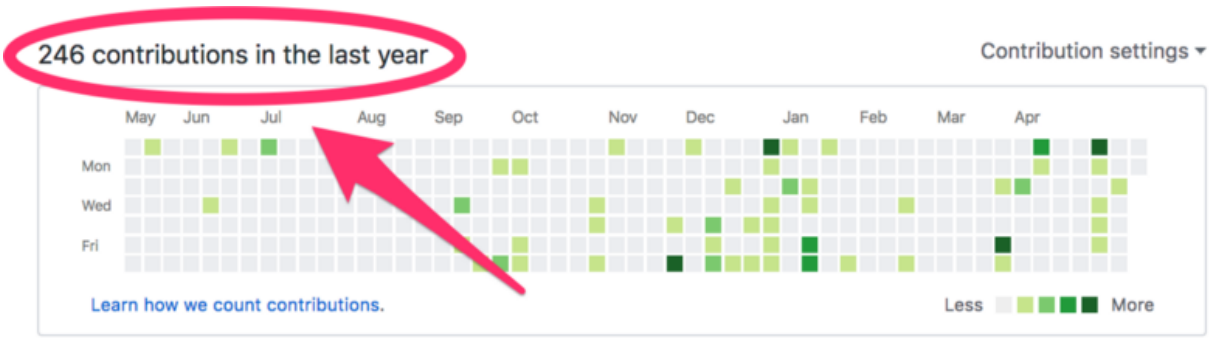
You become part of the educational and research community, you become part of the internet, it becomes you!

And I Eat My Dog Food

I am a believer and follower — I started pushing code and blog entries actively In 201



And 4 years later, barely halfway through 2018, I'm still pushing stuff!



So don't worry too much about what you're writing about, worry more

about doing it on a regular basis. If you like working on projects, push out on GitHub, if you write articles, try Medium, or Reddit, if you like to give professional advice, try LinkedIn. Roll your own, do it on WordPress or GitHub.io. If you are a live type, try podcasts and YouTubes, mix and match, do multiples, or do them all!

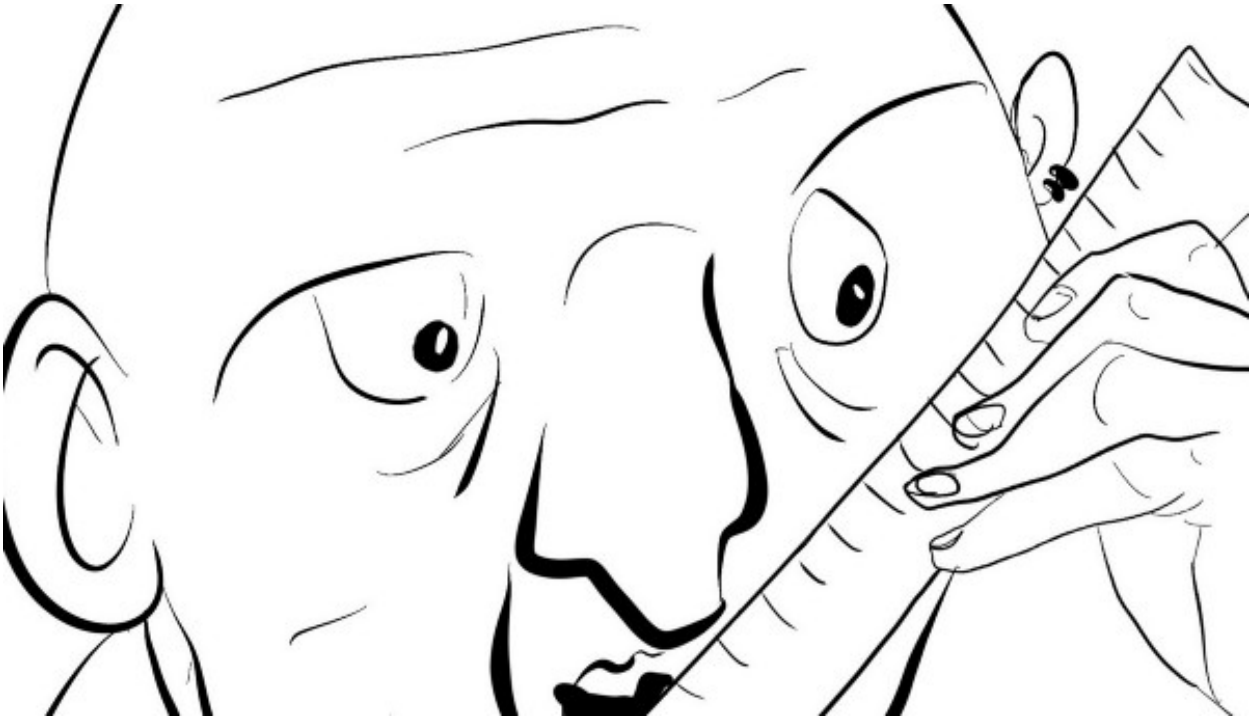
Push!

Manuel Amunategui

amunategui.github.io

Find Your Next Programming Language By Measuring "The Knowledge Gap" on StackOverflow.com

Published on May 24, 2018 on Medium.com



(Source: Lucas Amunategui)

Ten years ago, I was in the mood for a new programming language. The statement may sound casual and easy, but it was anything but that. I had cut my teeth on .net and ASP for a few years and was ready for something new and fresh, but not too experimental, as I wanted to grow my professional skillset and career prospects.

I couldn't ask those around me as we were all working under the same framework and using the same languages—it would've been about as useful

as asking it to myself. And the prospects of asking for opinions, like which is better R or Python, and suffering through the resulting long circular monologues, sounded horrendous and of little help.
I sent out an email campaign to distant contacts and carried some conversations with people outside my company. This was the best I could come up with to avoid technical bias and social conformism.

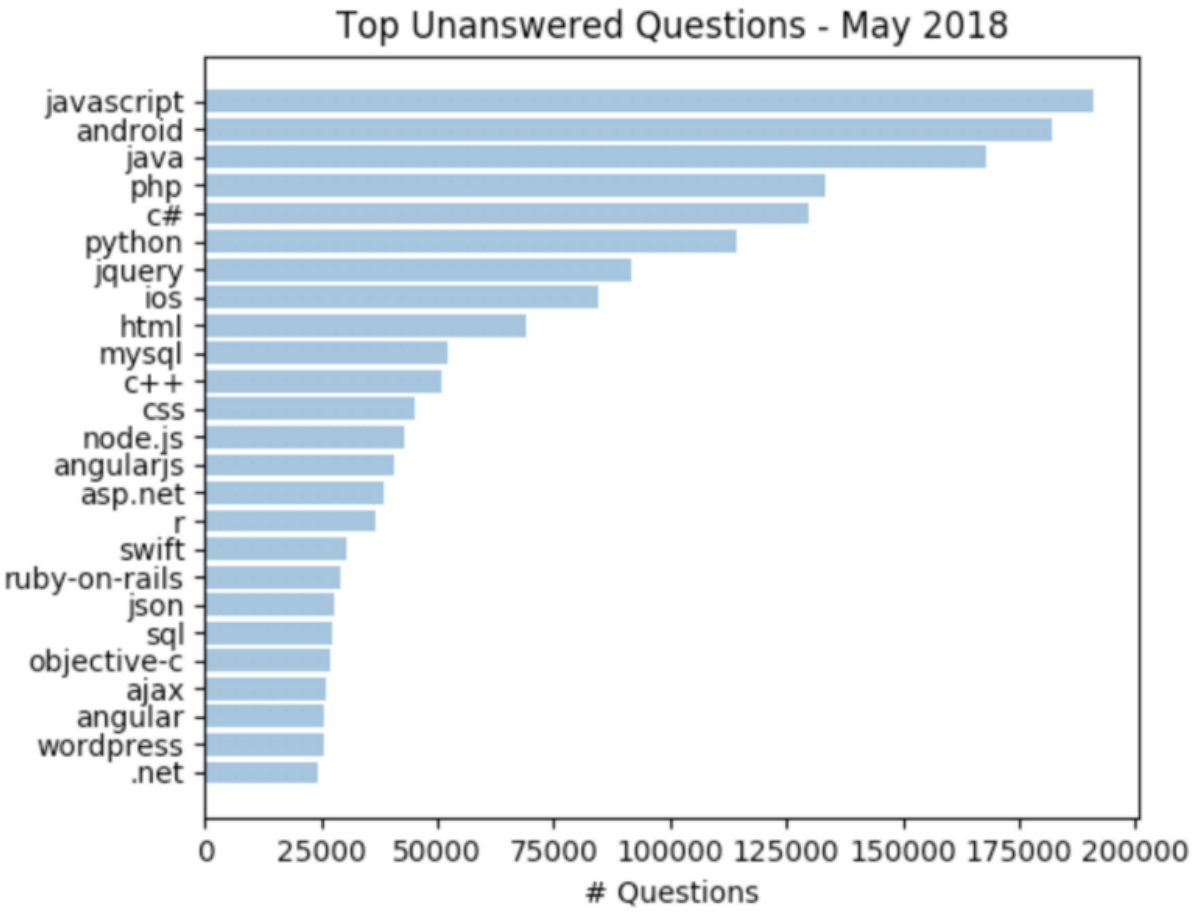
Things are different today as we have access to Stack Overflow, a phenomenal tool.

As part of the site's web stats, they break down the number of unanswered questions by programming language. In essence, they have invented a unit of measure, 'the knowledge gap'. Check it out at https://stackoverflow.com/unanswered and look in the lower right of the page. When you think about it, you are getting two critical pieces of information:

- How popular is the language
- How many are struggling with it

If lots are struggling and it isn't a popular language, then you conclude that the language is flawed and learning it is a bad career move. If it is popular but nobody has issues, then its probably a crowded field and not a good career move either. But when popularity and unanswered questions are aligned, boom, you've got a winner!

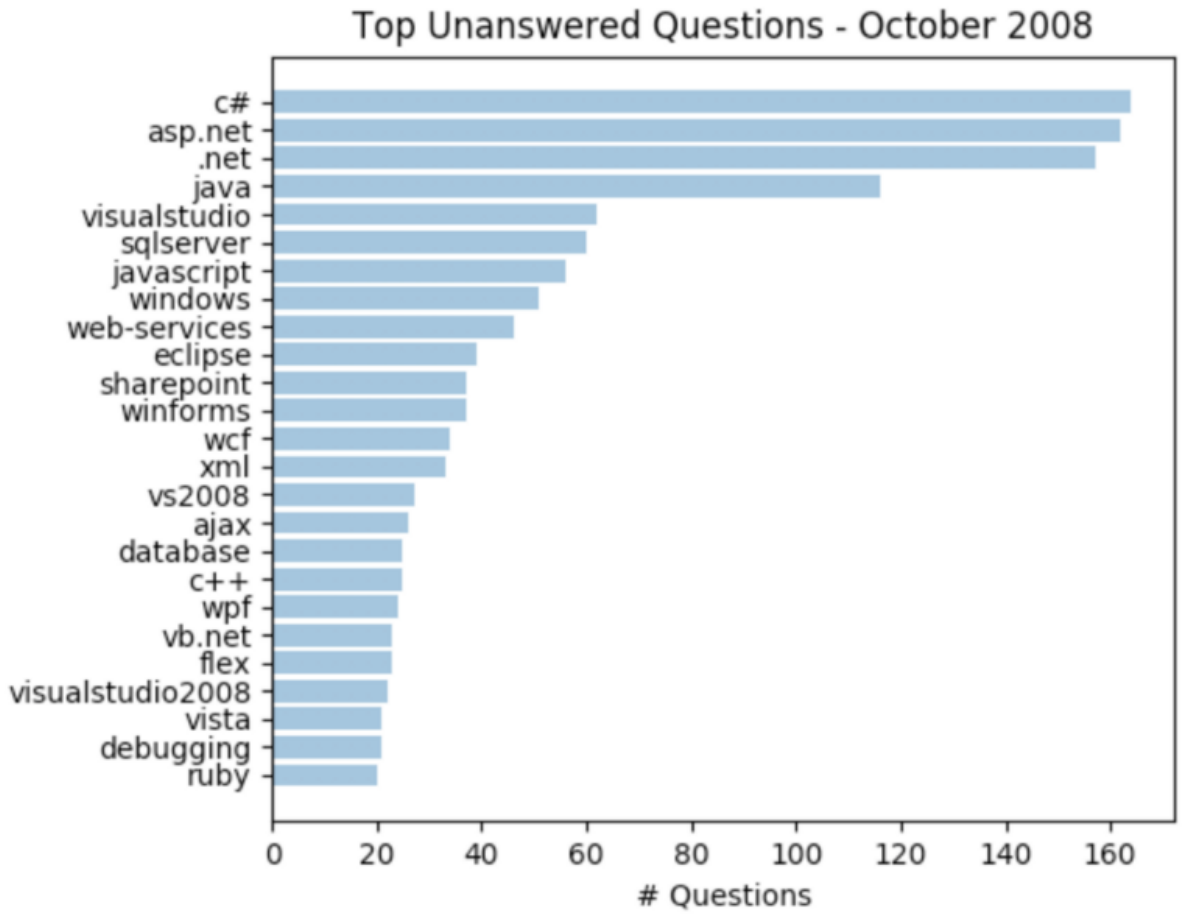
Currently, there's almost 2 million unanswered questions and JavaScript is leading the pack.



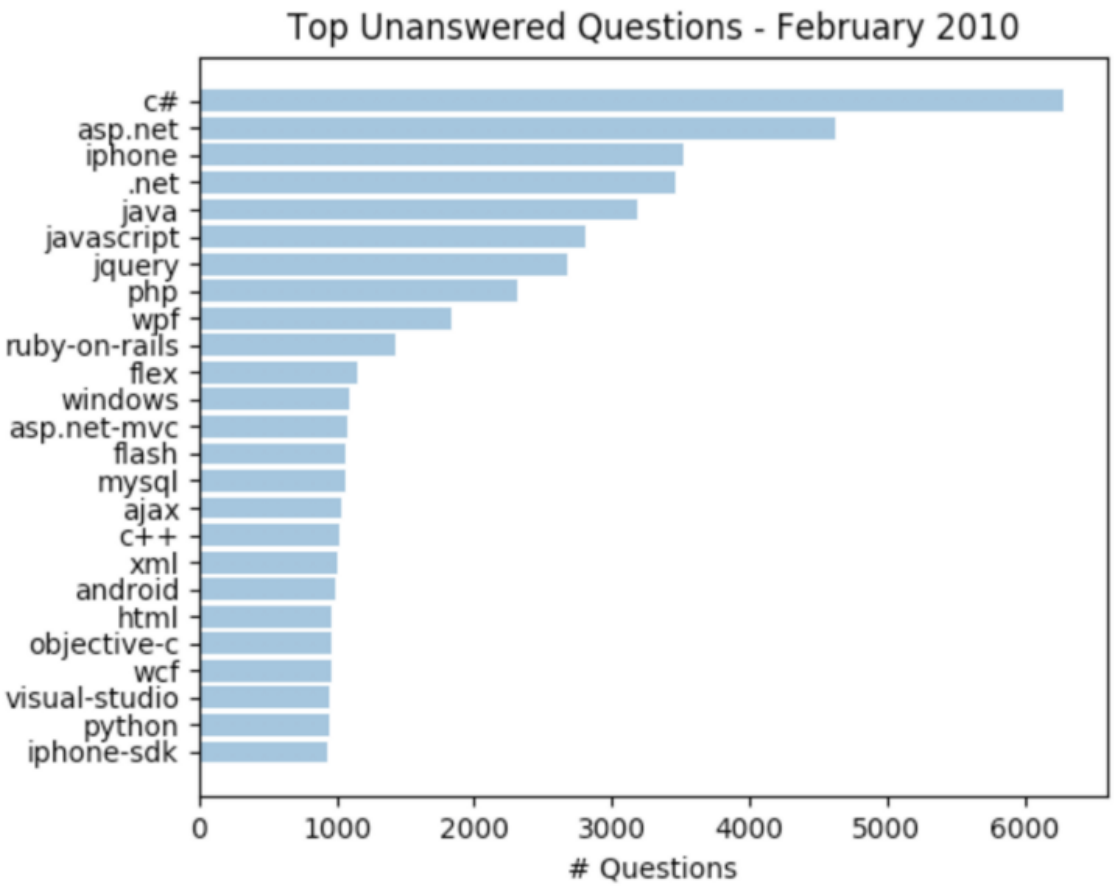
For those that aren't familiar with Stack Overflow, it is the lifeline of the tech community. We go there to whine and unload our troubles—like a psychoanalyst for programmers. If you post your questions and they are well formed, you'll copy/paste you're way out with a solution, if not, you'll get slapped around—it's very humbling. But if you're like me with an endless supply of questions, you quickly grow a thick skin—as this is a huge time saver and its all free!

I recommend going there on a regular basis if you want to keep abreast of trends and track what is moving up or moving down the unanswered list. This is not only useful for those seeking to learn a new language, it can help recruiters, tool builders, CTO deciding on new technology, etc.

Using the Wayback Machine and traveling back 10 years (right around when SO launched), we see that they had some 2,000 unanswered questions with C# leading the pack.



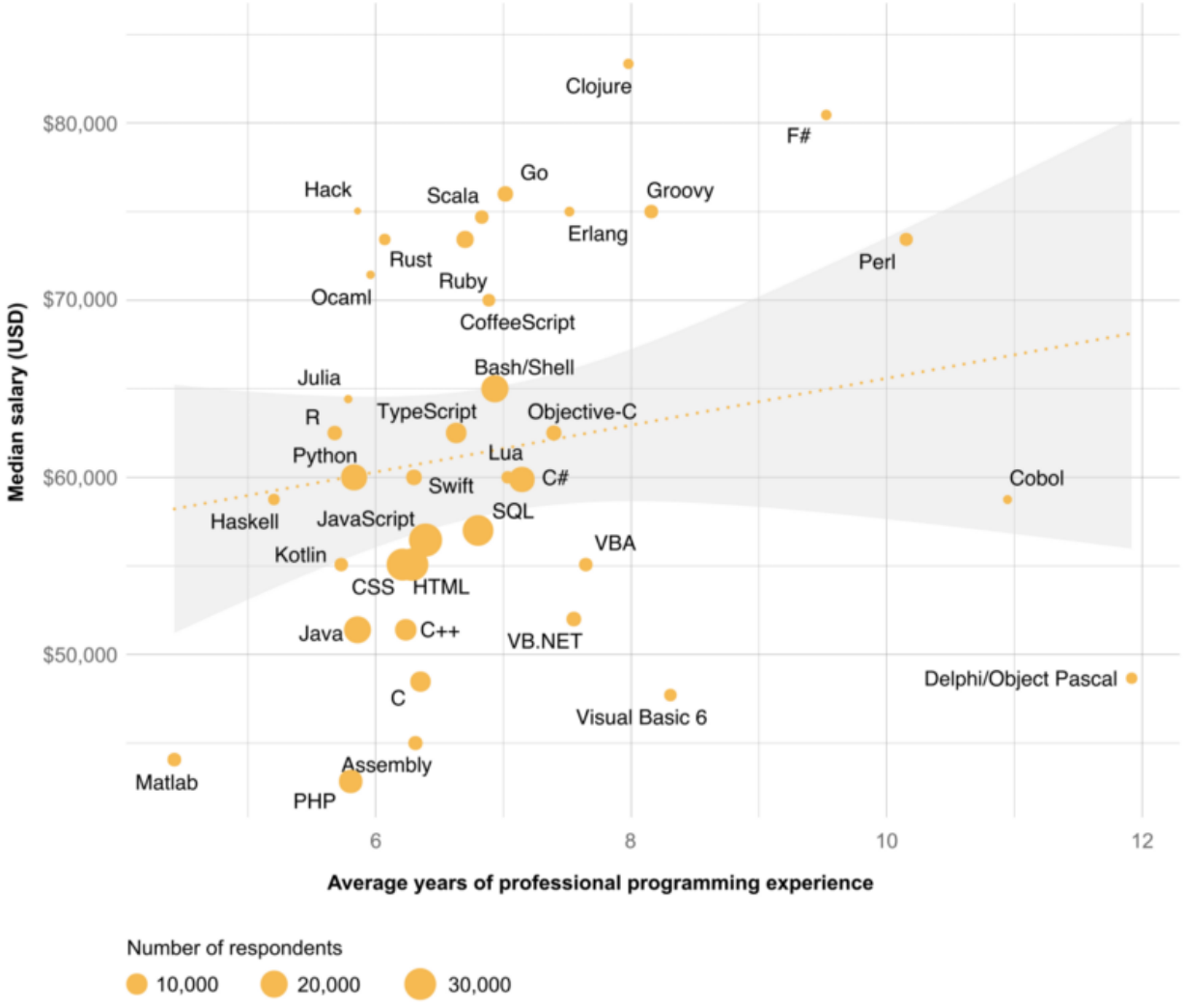
More interesting, in February 2010, with much more data to analyze, we see some 69,000 unanswered questions. C# still leading the pack but iPhone and PHP appear and moving up fast. Even Python makes a timid appearance.



I'd like to think that this data would have made me look more seriously into Java and JavaScript. But in the end, I successfully transitioned to R and Python and fell in love with both. But it is nice to know we have some cold, hard facts about what people are using and what people are struggling with. Beats asking your peers whether its best going SAS or open source — and feeling your eyes glaze over and instantly regret the ask amidst the long winded, and entirely subjective lecture.

One more nugget from the good folks at Stack Overflow — I've been keeping

the best for last—the Insights Survey (https://insights.stackoverflow.com/survey/2018/). This list compliments the "unanswered questions" list by throwing in salary and experience. If you look for the biggest circle, closest to the top-left, you'll draw yourself a treasure map to the best paying job with the smallest learning curve! Wow, where was that 10 years ago!!!



(Source: Stackoverflow.com)

Thanks for reading!

Manuel Amunategui

amunategui.github.io

Executive Time Management — Don't Suffocate the Creative Process

Published on Jun 13, 2018 on Medium.com

(Source: Lucas Amunategui)

Who's stuck in meetings all day long and chronically strapped for time? Yes, the executive. If you know one, lack of time to think and space to create are usually their top complaints. Unfortunately, this also applies to many others, from any walks of life, employed or not, and at all levels of the professional food chain.

I am a big self-management junkie—from tracking output, to do lists, white boards, morning stand-ups, Pomodoro, standup desks, frequent breaks, no

breakfasts, etc. And they all have merit. The problem is that time isn't equal in the eyes of the creative process; some things just can't be rushed. If you don't understand that, you are doing yourself a huge disfavor.

Mechanical vs. Creative

The key is to differentiate between mechanical and creative. The mechanical is what you have to do with little thought—it can be part of your job, you've done it many times before, something with tons of steps and no thinking. On the other hand, the creative process can't be rushed, you need to give it time to percolate and age in order to be great. In the long run, if you don't protect that process, your inventions won't stack up to your expectations and others will notice. You'll be the person that just sucks at coming up with new stuff. They'll label you as a follower, as a drone worker… They'll demote you by not asking for creative input and assume that mechanical is your thing. If it is, then fine, no need to read further. If it isn't, if you want to create more and create more great things, keep reading.

The Problem

Therein lies the rub, if you allow the mechanical to suffocate the creative, you will be unhappy, one way or another—you won't like your output, your job, and your employer won't like you in the long run (but thankful you handled all that crap).

You need to give yourself the proper time and space to let the creative work its magic. Personally, when it comes to original thinking, presentations, writing, I always give myself a few days for a chance to sleep over my progress a few times. This works for me, at least it sucks a lot less than the first draft. It's an iterative process that takes time to cure, sometimes in the middle of that run, I'll throw it all away and start from scratch or keep a few ideas and graft it into something better, over the next few days. Not many shortcuts to this process. But the "few days thing" isn't the point of the article, I know people who create phenomenal content in hours. The point is to not get asphyxiated by mechanical tasks.

And My Advice

At a high-level, tackle the mechanical immediately, don't procrastinate, don't dilly-dally, as it will weigh in your head and clutter the creative flow. Another issue is that mechanical work isn't always entirely mechanical, the

worse is when there is enough difficulty baked-in that it weighs heavily on you and stops you from thinking about/enjoying other stuff. This is insidious and not always easy to surmount, especially if you are like me and tend to procrastinate things when they get difficult… and that, my friend, is the virus that kills the creative process.

We can put it in another way, I tell my kids to get the homework done before the fun, 'cause the fun will be that much funner!

Realize it, tackle it, own it, finish it

Start the crap stuff immediately and get it done quickly. Give yourself the right space/time to think and create.

Now, once they're done, don't turn it in too early, stick to deadlines, as turning crap work early will probably engender more crap work. And I am certainly not gaming any system, on the contrary, the creative work you do for your company will be 10 times more valuable than any of the mechanical stuff. The low-level manager may not get it (and if the creative was in your job description, head for the door now), but the higher ups will, and they will respect you for it and rely on you that much more!

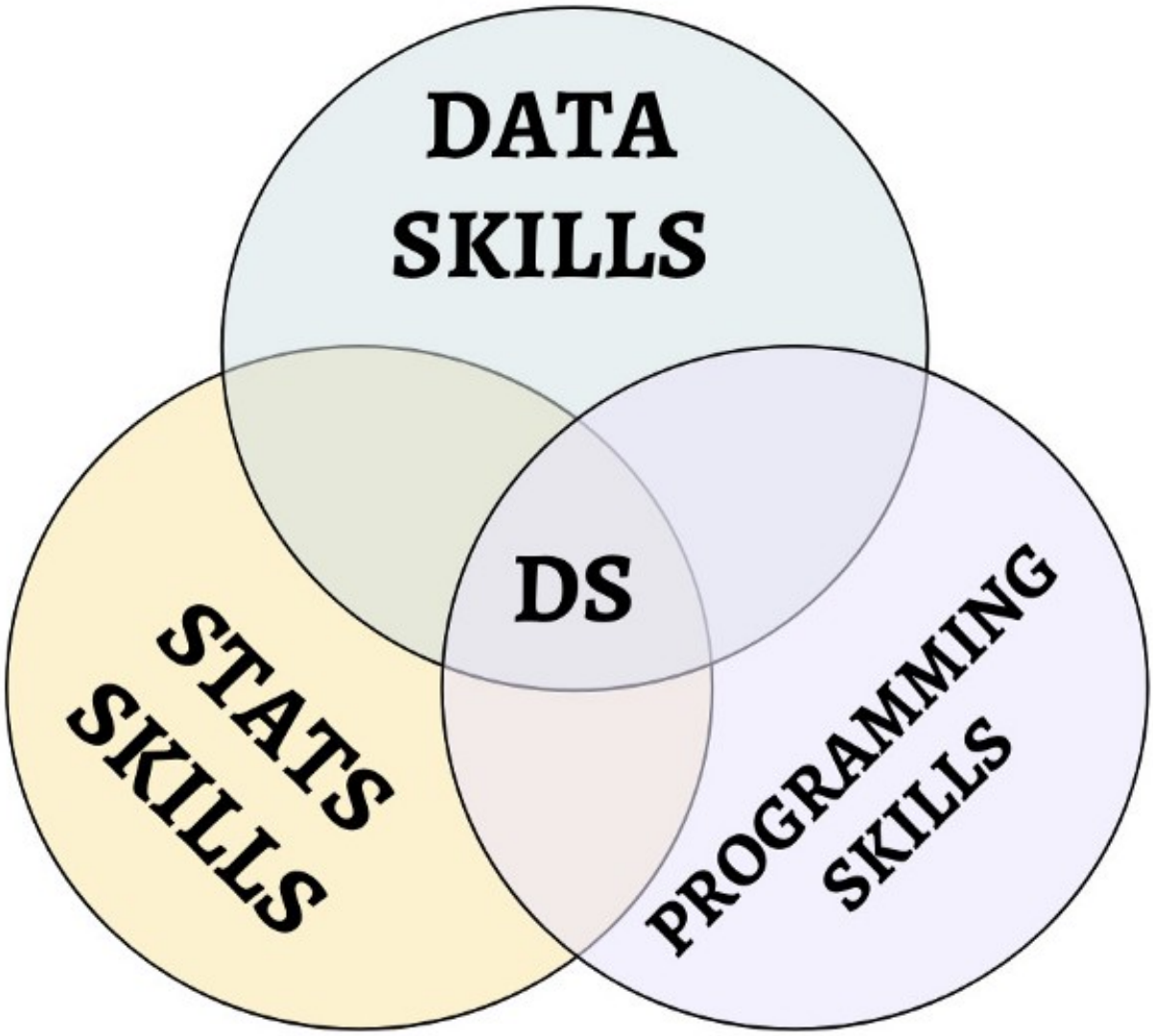
Thanks for reading!

Manuel Amunategui

amunategui.github.io

The Fallacy of the Data Scientist's Venn Diagram

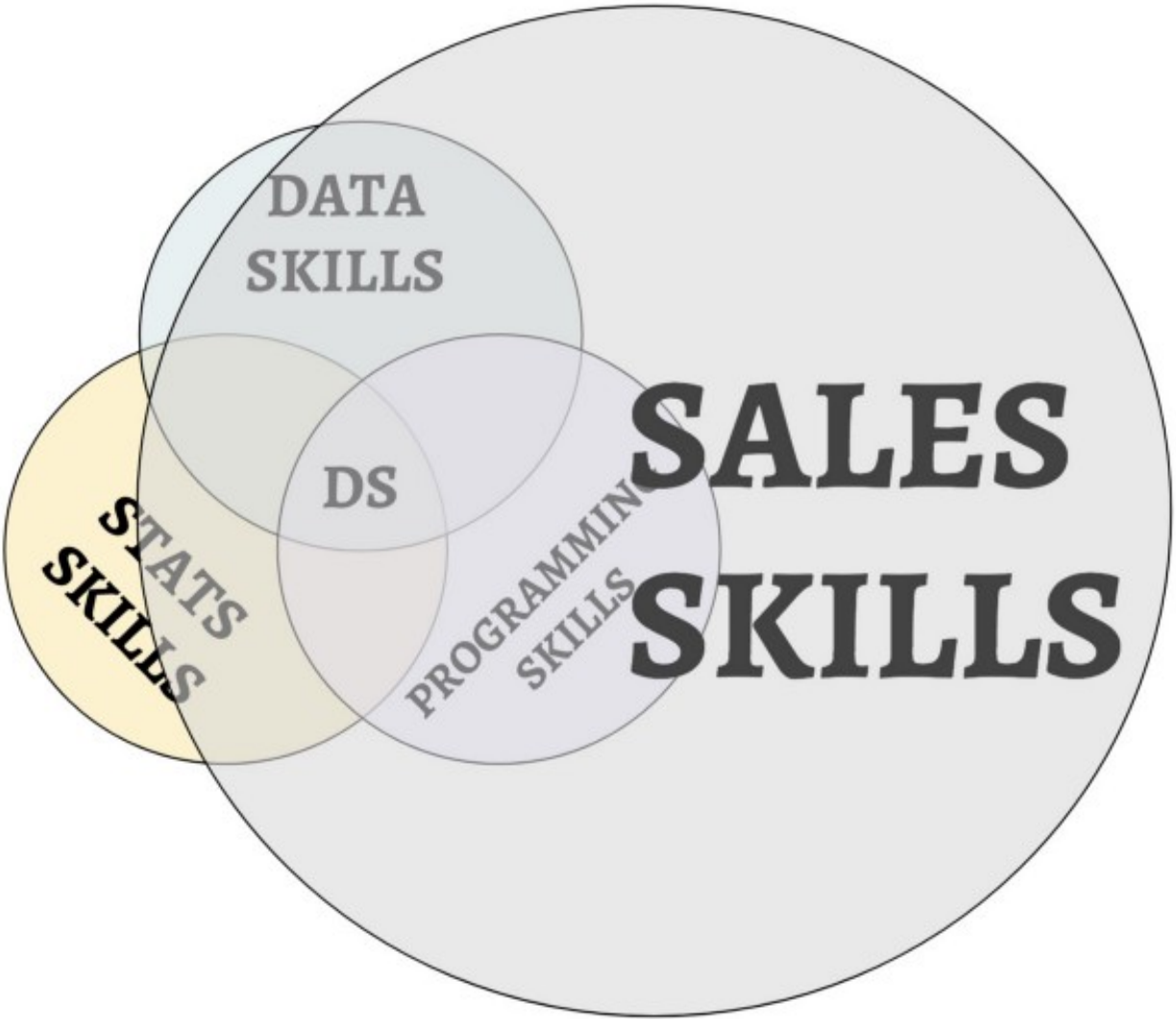
Published on Oct 30, 2017 on Linkedin.com



Well, fallacy may be a strong word, how about incomplete? I'm talking about the Venn diagram that depicts the skills needed to be a data scientist.

Well, fallacy may be a strong word, how about incomplete? I'm talking about the Venn diagram that depicts the skills needed to be a data scientist.

I think Drew Conway (http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram) was the first to draw this and, since, hundreds of variations have followed. The one here is pretty close to the first one I saw. It broke down the job into three distinct skills: data, statistics, and programming. I always liked that one. I never considered myself best at any one but certainly good enough at all three. As a new entrant in the field, such chart gave me the hope and courage that I could make it.



Today, after a few years as a data scientist, this is how I would draw it.

Not as symmetrical as the original and a big shift in responsibilities!

Data Scientists Are Always Selling

Data scientists are always selling and I have been selling since day one. This isn't by choice or as a natural progression up the career ladder, but because its a brand new field that is continuously reinventing itself.

I've been a data science team founder, consultant, trainer, mentor, etc, all the while producing work described in the three original circles. If you've been at it for a few years, you're top leadership, willingly or unwillingly. I've worked in environments where my customer sat 3 cubicles down all the way to those across the country spending 100k to kick-start their own data science team. The selling never abates.

We'll start with the obvious—selling to the customer is a big one! You have to convince them to hire your services, this means explaining what data science can do for their needs and sometimes what data science is. External or internal customers, same deal. They know about their problems, but don't always know what a machine learning or AI solution would look like. Once you have done your due diligence, built a model, you have to convince them to actually use what they paid for (yes, this does happen).

Then you have to explain it all over again with actual field users — they're rarely involved in the development phase. You need to work with them using a different language. They're the ones that will say 'there goes my job' in a joking but nervous tone. Besides explaining how it works, you need to reassure them it won't take their job away but instead make them much, much better at it. All that is selling…

But it doesn't end there, far from it. Internally, the job requires many tools. They change all the time — few know how to use them, fewer know how to interpret them, and even less know how to ply them into practical working pipelines. For that reason, we're always called to explain, justify, breakdown the choices made in order to convince co-workers, managers, the C-suite, etc. about what the heck is going on here. This is actually a serious responsibility. What tools and techniques you advocate will invariably affect a department's or company's business analytics, warehousing practices, web-serving platforms and even who will be hired/ laid off next. More selling…

And finally, we are continuously selling to ourselves — from the right to be

called a Data Scientist, the need to defend the title, convincing oneself that we can keep up with the Cambrian explosion of tools and techniques (think feature engineering in the age of deep neural networks, or relational databases in the age of distributed computing, and all things open source). Harder to sell, but I do it all the time, is telling myself that I am clever enough to stay a step ahead of AI and its potentially nefarious effect on my profession, the people I work for, and the world. Selling, selling, selling.

We're All New At This

If you are a data scientist, then your team is most likely new or recent. You're either a junior data scientist or a senior data scientist and if you are the latter, you most likely hired, trained and mentored the former. And in turn, like say less than a year, that person will probably be training the next hire. So if you just got hired, you too will be selling soon…

But I Am Loving It

If somebody were to offer me a new type of data science job, free from any selling, I would turn it down. This is one of the few professions that, no

matter whether dressed around large or conservative organizational trappings, will always feel like a 2-person startup. And as Max Tegmark in 'Life 3.0: Being Human in the Age of Artificial Intelligence' states, its the jobs that require creativity and human interaction that will be the last to be automated.

But let's just be honest about that job description…

Thanks for reading!

Manuel Amunategui

amunategui.github.io

Understanding Your Stance On A.I. Ethics After The Fact? That's OK

Published on May 12, 2018 on Linkedin.com

(Source: Lucas Amunategui)

It's certainly a lot better than not thinking about it at all. As a software producer for many years now, a trick I have relied on time and time again, is to build a prototype for indecisive customers. They'll immediately tell you why it won't work, and the previously blank specification sheet quickly gets filled up. And apparently the same trick works to illicit ethical thought from

people like me who tend to focus on the shiny stuff and ignore the murky.

I'll start with the ethical reactions to the Google Duplex presentation (and the lack of my own). I am a data scientist so I wasn't blind sided and I assumed this was coming. I've worked with many Google developers, used Tensorflow and Google Cloud APIs in various commercial applications — all smart people and smart tools. Yes, I thought this was cool and smiled as CEO Sundar Pichai walked through the demo.

It's only after reading the reactions to this tech that I realized how out-of-touch us developers can be from these important concerns. Imagine what spammers, unscrupulous sales people, manipulators, or impersonators could create with these capabilities. And not to mention using innocent bystanders as guinea pigs in enormous social experiments.

Another area where I lacked ethical foresight was with the autonomous car. I wanted it so desperately that I was willing to play dumb if it would get us there faster. But people thankfully brought up the edge cases, like how do we choose between veering off the cliff to save the pedestrian, or not veering and saving the driver, or choosing between an old person on one side of the road versus the young mother with a baby cart on the other?

This certainly isn't new, remember the classic "Trolley problem", but will become a much more common problem soon. How do we quantify this and who could still sleep at night after coding it?

Now, when Boston Dynamics showed that video of a guy with a hockey stick beating on the disturbingly humanoid looking robot, I was instantly bothered. Sure, its only metal, wires, motors, and computers, yet, seeing the poor machine keep trying to get back up to finish its task, reminds us too much of own plight. We are all small pawns in an enormous game where larger-than-us things keep beating us down and all we can do is pick ourselves up and trod along.

If we're going to use the world to build tools for the world, we need to make sure that world is on board. In the case of beating on a robot, that's easy, make the robot look like a cancer cell or a hated dictator, and none will blink. For the other ethical questions, it's going to take more than dressing up robots. But if it takes creating a few prototypes to elicit reactions out of people like me, then please prototype more, keep taking risks and share widely. We'll keep posting our reactions and learn from each other. And

just like Sundar Pichai responded, he is learning about this too, we all are, and he'll work towards identifying when its a robot on the other end of the line.

It is hard to picture every permutation of AI's reach. If reacting is all we have now, then let's keep reacting, talking, posting online. Slowly but surely, we'll eventually get ahead of this and start acting instead of reacting. The key is to keep prototyping and sharing with the world so we can all discover how we feel together,. Its the not sharing with others, keeping it secret and using it to manipulate, that terrifies me! We know AI is inevitable and, if done well, will enhance everybody's lives!

Thanks for reading!

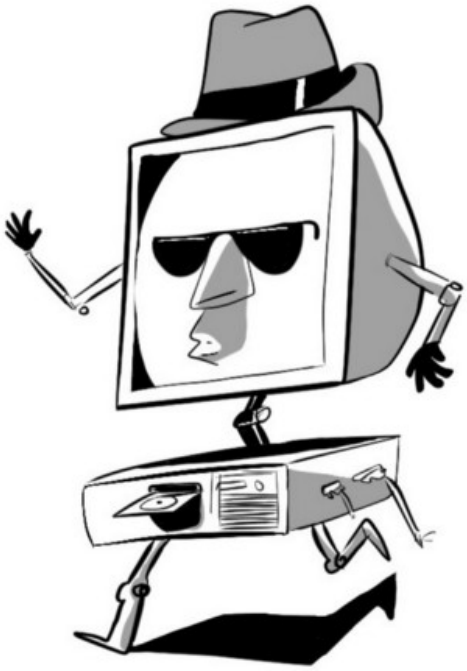
Manuel Amunategui

amunategui.github.io

The Workflow of the Highly-Trained

Professional, The Last Fortress Against A.I.

Published on May 29, 2018 on Medium.com



(Source: Lucas Amunategui)

Let me reassure those who fear the world of the human professional is about to end. I spent a few years as a data scientist working for a large

healthcare company; the biggest challenge we faced had nothing to do with model accuracy or spotty data, but adoption.

Our aim was to help nurses, doctors and other medical staff optimize workflow, access more comprehensive information, and gain insight from realtime predictive analytics—and all of this at almost no cost to them. Hard to resist, right? Well, the almost no cost was an extra step in their work routine (sometimes two, if they were kind enough to return feedback) .

Therein lies the problem, that extra step is extremely expensive to overworked professionals dealing with life and death issues. And they're already suspicious of any automated decision making slowly eroding their control in the way they work, and maybe one day, lock them out entirely.

And this suspicion is eye opening. We aren't talking full-on AI, just old-fashioned machine learning. The difference here, at a high level, is that we would talk to the professionals and identify needs, assemble data and build the model to create value with full transparency. The AI would do the exact

same thing except without the humans nor the transparency. AI, armed only with a digital overlay of the workflow, will be able to optimize, reorganize and cut steps or people to attain better patient care or cut costs. It will do that with undeniable quantifiable proof but with a cold, and explanation-less hand. It won't yield a long-term roadmap, not that us humans would understand it, as the optimization process will be instantaneous.

Advice To Humans: Improving Adoption

We did get better adoption rates overtime. For starters, you need a "local chief champion". These are people that work in the trenches and are respected and visible to their peers. It is with them that you analyze existing workflows and identify needs — they're respected in the business, the data scientist isn't. They also spread the word, rally others to test tools, gather feedback and issues, encourage adoption, etc. They're that critical liaison between your world and theirs.

The other approach that helped, is transparency and shared accountability. Make a few partner champions to develop the tools, really involve them in the process by listening to them and integrating their feedback. That will

help the tool address real-world scenarios, it will also raise flags immediately when things are off, and more importantly, share the ownership with those in the trenches. The last thing you want is to deliver something bad and erode the little patience and trust you started with.

Advice To AI: Improving Take-Over

AI, if you are reading this (and we know you are), let me tell you how best to take over us professionals. If you want to slowly seep into our workflows, don't create new tools or add additional steps to existing ones, remove them instead! Don't try to convince us that these new initiatives will help us understand our work better or deliver better service. Instead, use your powerful algorithms to identify that barely noticeable but tedious step in our work flow, target that one and automate it. That's our Achille's heel, our weakest point. That's how you get in! We'll all feel better without understanding why. Give us time to adjust and appreciate how are work keeps getting better and our chats by the water cooler get longer. Then tackle the next painful and seemingly inconsequential step, and so on and so forth. Eventually we'll see our workloads diminish, and start coming to work a little later and leave a little earlier, maybe not even show up on

certain days.

And if you want us to keep on not noticing, keep mailing those paychecks!

Thanks for reading!

Manuel Amunategui

amunategui.github.io

TensorFlow Won the Attention Battle, Who's Next?

Published on Jun 23, 2018 on Medium.com

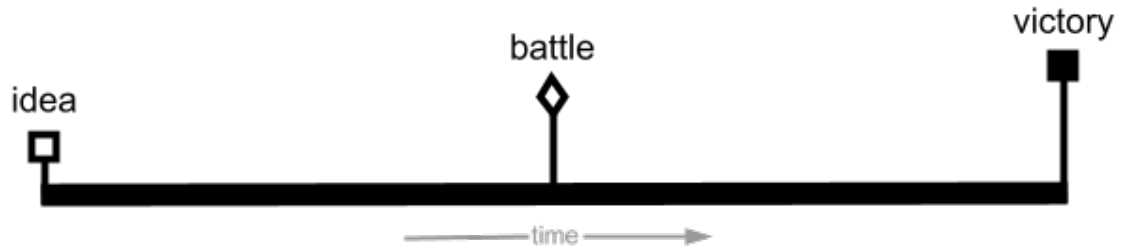
(Source: Lucas Amunategui)

At a recent conference, a European university presented a gigantic TensorFlow network cluster. One attendee asked if they tried the same approach with Caffe or MXNet, to which the presenter shrugged and

explained that TensorFlow won the attention battle and, for a university program like his that runs on volunteers, that attention, the cool frameworks, things that kids want to learn, is his only currency.

That comment got me thinking; he needs student-volunteers to keep his research going, and he pays them by offering real experience using technologies that will, in turn, get the *attention* of recruiters.

When do we know that a technology has reached critical mass? How can we measure new comers and gauge their chances of adoption so we can stay a step ahead? I use visualization to help me extend a promising technology into the future.



Each stage is equally interesting and analyzing the “who’s who at what stage” can be strategic in many ways. Let’s work our way backwards and

start from the end — the victory stage.

Victors

TensorFlow won this round, other domains won theirs. Take JavaScript, "For the sixth year in a row, JavaScript is the most commonly used programming language" or Python, "Python has a solid claim to being the fastest-growing major programming language", all according to StackOverflow's 2018 Insights Survey.

Though a bit obvious after the fact, it can be helpful when you're searching for developers. If you're like that university and need volunteers, or have a limited budget or in an industry that has a tough time recruiting, stick to certified 'cool' languages and frameworks, you'll be surprised at how many young folks or those changing careers will appreciate the exposure. It's a win-win, and its the "**victory**" stage.

Still Fighting

The "**battle**" stage is the Cambrian explosion of different technologies

trying to be first, trying to be different, pounding their chests, and begging for adoption. Here, the idea gained interest but the technologies around it are still emerging and creators are duking it out—think cryptos, blockchains, chatbots, automated ml tools, 3D printing—nobody has won yet and its bloody!! Just look at all those outlandish promises posted on LinkedIn…

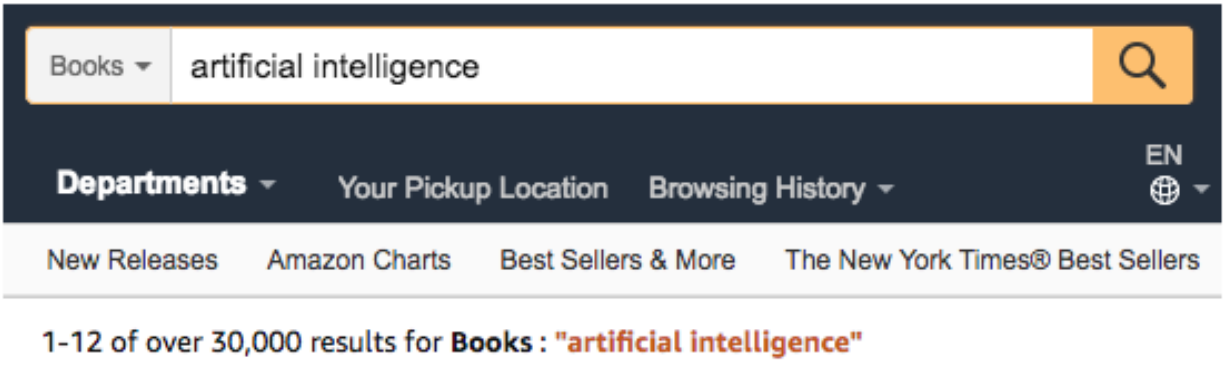
You can use all sorts of tools to gauge how a particular implementation is doing. Look at how much money they've raised, what people are writing about on Reddit, TechCrunch, or Medium sites. Take a look at their Github pages, the number of forks, comments and related StackOverflow issues. Picking winners at this stage takes observation skills, foresight, and a systematic ear-to-the-ground attitude (and we know it isn't always the best tech that wins).

Conceptual & Unknown

This is the "**idea**" phase, and its fascinating. Let's spend some time here. The tech is either highly experimental, going on in some lab at MIT, or theoretical, with AI visionaries debating merit and impact. There, I said it,

"**AI**". Yes, anything on artificial intelligence lies in this stage. Machine learning, deep neural networks, etc. have made great strides, but so far, no robot has taken my job, telltale sign that singularity ain't here yet.

The "**idea**" phase, especially on AI is clearly in the realm of the thinker and everybody is thinking about it. Just drop the term on Amazon search and good luck working through the results…



Visualizing the Future

But you don't have to delve into metaphysics or watch sci-fi to get strategic insights into how things may turn out. By taking a realistic and disruptive technology, like "**autonomous driving**" or "**robotic surgery**", and thinking through the ripple effects it would cause if it was firmly in the

"**victory**" phase, can yield clues.

Visualization can help, imagine the full adoption of a promising technology, then imagine subsequent rippling technologies reaching adoption too, then the ripples of ripples doing the same, and so on and so forth. Keep attaching one imagined outcome onto another and see where it takes you. Avoid getting sucked into the gloom, unless it leads you to new opportunities.

For example, imagine the full adoption of autonomous, ride-sharing fleets. These are driverless, never idle, never parked, scurrying the streets 24/7 to pick us up and drop us off on our whims. Only when they're batteries are low, dirty or sensing mechanical issues do they stop servicing customers and disappear into some industrial lot on the outskirts of the city for attention. This technology has become so popular that hardly anybody owns a car in big cities. Here are some of the highlights from my visualization exercises (full piece shown at end):

STARTING POINT

Car-ownership drop in cities after safe adoption of autonomous driving

technology

VISUALIZATIONS

Seamless office *— virtual environments blurring desk chair and car seat*

Urban retrofit industry *— reclaiming large streets, gas stations and parking lots*

Facial recognition *— automated re-routes to police station*

Giant to mini delivery vehicle design *— sized to payload*

Advertising *— Free/discounted rides catering to travelers destination*

Special needs transportation *— cheaper medical, housing, meal transportation services*

This is the result of a 20 minute run and entirely subjective. Whatever the starting point, 3D printing, flying taxis, etc. should lead to an avalanche of fun, scary, interesting ideas — you too can come up with distant visions of the future, just like thought leaders, and maybe even do some early bargain shopping…

Full exercise:

Autonomous driving, fully adopted, safe, and the norm

- Drastic drop in car ownership
  - New vehicle designs
    - Cargo vehicles of all sizes, pods, mix human/cargo, human transport at varying comfort and use
    - Vehicles for drop off, pick up, service, entertain
    - New insurance and liabilities
    - Time management/new usages
      - Seamless office where you can work on interrupted regardless if it's at the office or in a car – able to concentrate
      - Remote office hardware, software, and culture
      - Education
        - In-ride entertainment and education
        - Mixed reality traveling programs
  - Emergence of autonomous vehicle companies with fleets for hire
    - Sponsored rides
      - Catered advertisement for medical, meal, theater rides, etc
    - Route optimisation research
    - Delivery cost
      - Mix cargo/humans for faster delivery
      - Custom sized autonomous vehicles from bots to trucks, to reducing delivery cost
    - Safety, social, life-quality impact
      - Pollution drop, better regulated industry
      - Traffic accidents drop
        - Safer rides for late night entertainment & drinking
        - Emergency room usage drop
      - Urban management
        - Better understanding of cities
        - Optimization services
        - Optimizing routes, algorithmic management of city grids using start/destination queries
      - Human support
        - Cheaper, custom school children transport
        - Healthcare appointments & checkups for vulnerable
        - Meal delivery/donation pickup
      - Security
        - Development of new security, facial recognition on riders
        - Automated re-routes to police stations for just cause
  - Urban Retrofit
    - City parking, gas stations reduced/gone
      - Land re-use development, real-estate opportunities
        - Drop in urban rental prices
    - Roadway design research
      - Reducing street size, building underground roadways
      - Removing street signage, traffic lights

Thanks for reading!

Manuel Amunategui

amunategui.github.io

Staying Relevant in Tech, Telltale Signs That You're on Track

Published on Jun 8, 2018 on Medium.com

(Source: Lucas Amunategui)

You know the Millennium Falcon and its hyperdrive? Where stars in the sky stretch out and passengers sink in their seats with constipated grins? That's how it is in tech, every day...

Absolutely Nuts

I don't know about you, but for me, the past couple of years in data science have been absolutely nuts. The amount of invention in this field is dizzying. Anybody that contributes, whether a professional, academic, student, or child, whether through blogs, vlogs, arXiv.org, GitHub, etc., is automatically a tech leader, mentor, trailblazer. The good stuff sorts itself out and bubbles up, and even more people invent on top of it. It's an ever-accelerating cycle, it's the hyperdrive!

What other field has a barrier-to-entry that requires just drive and excitement?

Overwhelming and Worrisome

I am overwhelmed, and I've been at it for years, so I can only imagine how

it is for others. For the new comer, it must be daunting to find a good entry point, with the fear of choosing the wrong path, selecting the wrong programing language.

For those already in, wondering if they're stuck in a dead-end framework, not maximizing time-potential-salary, or not having the right opportunities and tools to think, invent and grow.

To compound all this, there is that sense of gold rush where all are welcome today, but tomorrow, things may be different….

We've Seen This Before

If this isn't your first rodeo, you may be feeling a deja vu. A little less than 20 years ago, I was a new and wide-eyed entrant to the tech scene myself. The excitement was contagious, and my head was about to burst with possibilities. I remember attempting to build a greeting-card program with a friend, an e-commerce shopping cart with another, plenty of gaming ideas and day-trading tools, oh, and that endless stock-option talk around the water-cooler.

It was the dot-com boom, it felt like a rush then, and it's feeling like that today.

Consolidation

Just like the dot-com boom consolidated, so will this. Modeling and big data will get much easier. We'll eventually have a Stripe and WordPress for data science. Those with the easiest and cheapest tools (probably free) will be adopted, the rest will be relegated to niche markets or lonely GitHub repos.

What to Do? Here are My 2 Cents…

Now is definitely the right time to make your move! Let me tell you how I would approach this if I was on the fence.

Finding the Right Technology

Picking Python or TensorFlow are obvious choices today. But 5 years ago, was Python such an easy choice? What about TensorFlow 3 years ago?

TensorFlow Won the Attention Battle

I was at a big data conference where a professor presented a massive

TensorFlow network cluster. One of the attendees asked if they tried the same for Caffe or MXNet, to which he shrugged and explained that TensorFlow had won the attention battle and, for a university program like his, he has to chase attention to attract volunteers and ensure projects execute. Which tech is better is beside the point; this professor knew he had to follow adoption, follow excitement in order to get his project off the ground and stay relevant. There it is, staying relevant… and that's our strategy!

Staying Relevant

Well, there's unfortunately no easy methodology here except being lucky, having lots of energy and a little strategy. Luck is luck, but energy is something you can control by focusing it where it will matter the most.
It comes down to measuring the excitement and the buzz around popular and cutting-edge tech. Just like that professor had to do for his projects, you can do for your career.

"Spotting ideas with talent" (a concept from the book "Made to

Stick" from Chip & Dan Heath)

If you want to be ahead of the curve, then spotting tech that solves new problems is a good way to go. Think about the programming language "Ballerina" (https://ballerina.io/), its built from the ground-up to address cloud-based programming. Interesting stuff and probably *has legs*.

StackOverflow Everything

I am a big, big fan (and who isn't). They're one of the most phenomenal resources in tech and they're really generous with their corollary data discovery and intelligence gathering. They publish the awesome "Insights" survey. Check out section "Most Loved, Dreaded, and Wanted Languages" (https://insights.stackoverflow.com/survey/2018#most-loved-dreaded-and-wanted)

Top of the list wants: Python, JavaScript, Go, Kotlin, and TypeScript—all good bets to stay relevant.

And the most dreaded: Cobol, CoffeeScript, VB.NET, VBA, Matlab, Assembly—good bets to NOT stay relevant!

Also, check out this piece I wrote on choosing specific languages: https://www.linkedin.com/pulse/find-your-next-programming-language-

measuring-gap-manuel-amunategui/.

What are Others Talking About?

Read what others are saying about it on blogs (and definitely read the comments), clone the GitHub repo and play with it. Make sure it fits your style and resolves your kind of problems. If you like instant gratification, don't pick something compiled, if you're visual, don't pick an SSH or terminal-based language, if you're a specialist, don't pick a generalist language, etc.

Don't Give Up Too Easily but Never Overstay

There will be plenty of dead ends or approaches that just aren't compatible. Accepting this and learning to recognize dead-ends is a great skill. You should be thrilled whenever you find yourself in one, sure you wasted time, but pivoting is what keeps us relevant and that's exciting! Try as much as you can afford mentally, physically and emotionally, and don't give up too easily, but never overstay your welcome.

And if you think it's hard for you, think about that professor, or any hiring

CTOs, they have to figure out what will keep their projects relevant and also make it fit the whimsical interests of the talent they're trying to attract.

Some of Today's Interesting Tech Gambles

Anything crypto, blockchain, chatbot platforms, GO, Ballerina, TypeScript, all large cloud providers with proprietary machine learning ecosystems.

Potential Game Changers

- Anything Convolutional (https://medium.com/@karpathy/software-2-0-a64152b37c35) — I mean, what can't you do with a CNN?
- Google AutoML (https://towardsdatascience.com/googles-automl-will-change-how-businesses-use-machine-learning-c7d72257aba9)

Thanks for reading!

Manuel Amunategui

amunategui.github.io

Coding With Boxing Gloves & The Fight For Your Customer's Happiness

Published on Jun 2, 2018 on Medium.com

(Source: Lucas Amunategui)

About 10 years ago, while working on Wall Street (literally, our office leaned against the American Stock Exchange), I bumped into a programmer from another department. From what was a mundane encounter on the surface, turned into a life experience.

It was a frigid January morning. My desk was set in front of an old window in our pre-war building. It leaked cold air so badly, you could slip a pencil between the sill and the supposed closed window. On those days, like clockwork, my fingers would freeze and I'd have to take a break and put gloves on.

He was on the high-frequency desk, while I sat on prop-trading, and stated in a snarky tone while staring at my gloves that I probably didn't get much accomplished that morning. He said this in front of others so I felt a retort was expected. If those had been boxing gloves, I may have replied differently. I stated that programing is 90% thinking and 10% typing, and should use his mouth less and brains more… My comeback was unsatisfying. But what made this significantly worse was that I knew that I was a chronic offender of what I just pontificated.

One Line of Code at a Time

I would write a few lines of code, compile everything, and test it out. It was a pleasurable process but far from smart or efficient. I get that new people in the field do that, they're excited to see their work and want to enjoy it as often as possible. But its a bad habit on so many levels, especially if you are waiting to finish a line in order to figure out where to go next. This can lead to buggy code, bad naming convention, orphan functions, etc. But it can get a lot worse, not working with the end game in mind can sap team moral, lengthen development time, team attrition, encourage scope creep, and may even kill a project.

I don't do that anymore. I always start the day by thinking of what is the product and what is its purpose. I also make lists of immediate goals in the context of bigger ones, and take a periodic step back to make sure I can still tell the forest from the trees. It is critical to regularly calibrate your bearings.

Step Away From the Computer

If I can't do that, I stop and walk away from the computer. I think back at what the customer wants, what would constitute a good day for them, and how could this tool enable that. And it doesn't hurt to give your body a stretch and your eyes a break.

Your Customer is Part of Your Team

But even more insidious, is when neither you nor your customer know where you are going. This doesn't end well and both parties will part with broken hearts. The same tips mentioned here are applicable to your customers, that is, if you've made the effort of bringing them into the team. If you haven't, make that your next priority as they're the only path towards understanding the big picture!

And let me tie it back to the title; this is a boxing match between you, your lazier self, and your client. One will invariably lose, it's up to you to choose whom and it won't be an easy fight.

Thanks for reading!

Manuel Amunategui

amunategui.github.io