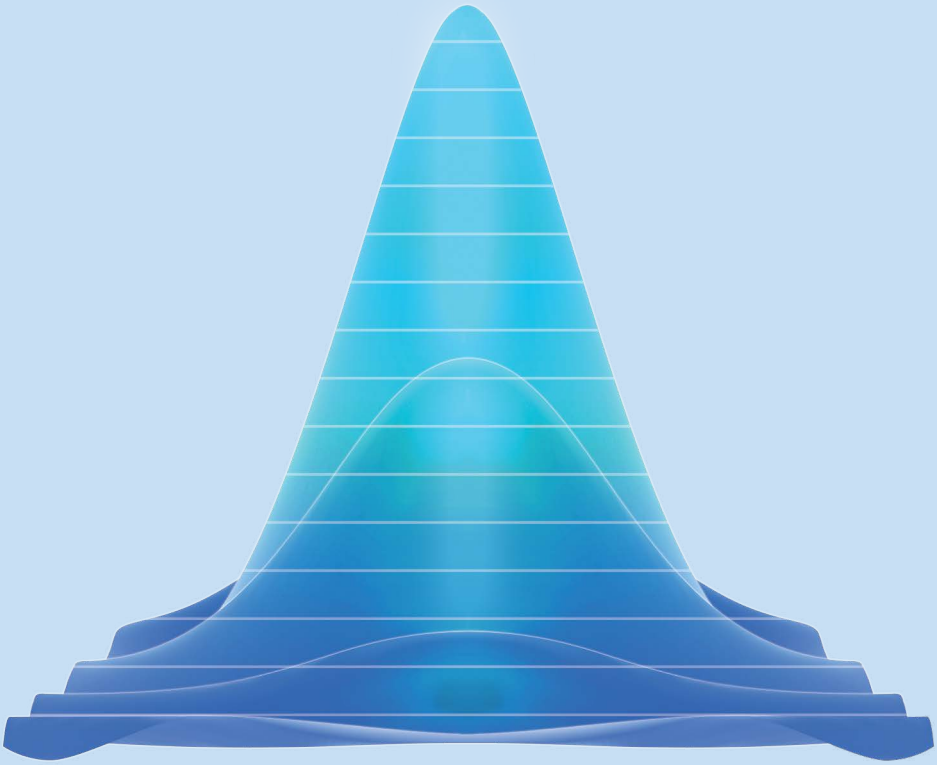
THE DATA SCIENCE HANDBOOK



ADVICE AND INSIGHTS FROM 25 AMAZING DATA SCIENTISTS

FOREWORD BY JAKE **KLAMKA**

DJ **Patil,** Hilary **Mason,** Pete **Skomoroch,** Riley **Newman,** Jonathan **Goldman,** Michael **Hochster,** George **Roumeliotis,** Kevin **Novak,** Jace **Kohlmeier,** Chris **Moody,** Erich **Owens,** Luis **Sanchez,** Eithon **Cadag,** Sean **Gourley,** Clare **Corthell,** Diane **Wu,** Joe **Blitzstein,** Josh **Wills,** Bradley **Voytek,** Michelangelo **D'Agostino,** Mike **Dewar,** Kunal **Punera,** William **Chen,** John **Foreman,** Drew **Conway**

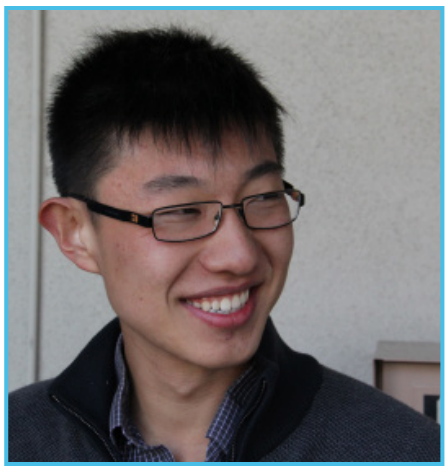
BY CARL **SHAN** HENRY **WANG** WILLIAM **CHEN** MAX **SONG**

CONTENTS

CONTENTS

CONTENTS

ABOUT THE AUTHORS



CARL SHAN *is a Data Science for Social Good Fellow in Chicago, where he works with President Obama's former Chief Scientist on applying machine learning and data science to pressing policy issues. An avid reader, he co-authored The Data Science Handbook to help bring stories and wisdom from pioneering data scientists into the lives of as many readers as possible. When not mired in data, Carl can be found at a pool table, or pretending to know the lyrics of the latest hit pop song.*

Carl holds an honors degree in Statistics from UC Berkeley.

HENRY WANG *is an investment analyst with New Zealand's sovereign wealth fund, where he focuses on private investments in alternative energy technologies. He is interested in the intersection of data science and traditional capital intensive industries, where data driven techniques can be used to better inform operational and investment decisions. Henry is a simple guy who enjoys simple things like travelling, reading, and making delicious instant ramen.*

Henry holds a Bachelors in Statistics from UC Berkeley.

WILLIAM CHEN *is a data scientist at Quora, where he helps grow and share the world's knowledge. He is also an avid writer on Quora, where he answers questions on data science, statistics, machine learning, probability, and more. William co-authored this book to share the stories of data scientists and help others who want to enter the profession. For fun, he enjoys speed-solving Rubik's cubes, building K'NEX ball machines, and breaking out from "escape rooms".*

William holds a Bachelors in Statistics and a Masters in Applied Mathematics from Harvard.

MAX SONG *is a data scientist currently working on secret projects in Paris. Previously, he was the youngest data scientist at DARPA-backed startup Ayasdi, where he used topological data analysis and machine learning to build predictive models. He co-authored the Data Science Handbook to share the the wisdom of pioneers for those looking to trailblaze their own data science journeys. When not feverishly coding, he can be found playing improv games and seeding an intellectual gathering (Salon) in far-flung corners of the world.*

At the time of writing, he is on leave from Applied Mathematics-Biology at Brown.

BRITTANY CHENG *is an Associate Product Manager at Yelp who recently launched Yelp in Taiwan. She created the layout design of The Data Science Handbook and has also worked on layout designs for 120 Data Science Interview Questions and The Product Manager Handbook. When she isn't designing handbooks, she likes to eat, talk about eating, drink tea, and rant about umbrellas. Read her Yelp reviews to see what she's been eating recently.*

Brittany holds a degree in Electrical Engineering and Computer Science from UC Berkeley.

To our family, friends and mentors.
Your support and encouragement is the fuel for our fire.

PREFACE

In the past five years, data science has gone from a nascent, tech industry competency to a field that is having a global, cross-industry impact in almost every major area of human endeavour. From education, to energy, to government, to non-profits and, of course, software and the Internet, data science is creating immense value for companies and organizations across the world. In fact, in early 2015, the President of the United States announced the creation of the new role of Chief Data Scientist to the White House, appointing one of the interviewees of this book, DJ Patil.

Like many innovations in the world, the birth and growth of this industry was started by a few motivated people. Over the last few years, they founded, developed and advocated for the value that data analytics can bring to every industry around the world. In The Data Science Handbook, you will have the opportunity to meet many of these founding data scientists, hear first hand accounts of the incredible journeys they took, and where they think the field is headed.

The road to becoming a data scientist is not always an easy one. When I tried to transition from experimental particle physics to industry, resources were few and far between. In fact, although a need for data science existed in companies, the job title had not been created yet. I spent a lot of time learning and teaching myself, working on various startup projects, and later saw many of my friends from academia run into the same challenges.

I saw a groundswell of incredibly gifted and highly trained researchers who were excited about moving into data-driven roles, yet they were missing key pieces of knowledge, and had trouble transferring the incredible quantitative and data analysis skills they had gained in their research to a career in industry. Meanwhile, having lived and worked in Silicon Valley, I also saw that there was a very strong demand from the technology companies who wanted to hire these people.

To help others bridge the gap between academia and industry, I founded the Insight Data Science Fellows Program in 2012. Insight is a training fellowship that helps quantitative PhDs transition from academia to industry. Over the last few years, we've helped hundreds of Insight Fellows, from fields like physics, computational biology, neuroscience, math, and engineering transition from a background in academia to become leading data scientists at companies like Facebook, Airbnb, LinkedIn, New York Times, Memorial Sloan Kettering Cancer Center and nearly a hundred other companies, with a strong alumni network on both the East and West Coast.

In my personal journey to enter the technology field, and creating a community for others to do the same, one key resource I found to be tremendously useful was conversations with others who had successfully made the transition themselves. As I developed Insight,

I have had the chance to engage with some of Silicon Valley's best data scientists who are mentors to the program:

Jonathan Goldman created one of the first data products at LinkedIn — People You May Know — which transformed the growth trajectory of the company. DJ Patil build and grew the data science team at LinkedIn into a powerhouse and in the process co-coined the term "Data Scientist." Riley Newman worked on developing product analytics that was instrumental in Airbnb's growth. Jace Kohlmeier led the data team at Khan Academy that helped to define how to optimize learning at a scale of millions of students.

Unfortunately, face-to-face time with people has trouble scaling. At Insight, to maintain an exceptional high quality and personal time with its mentors, we accept a small group of talented scientists and engineers three times per year. The Data Science Handbook provides readers with a way to have that in-depth conversation at scale. By reading the interviews in The Data Science Handbook, you will have the experience of learning from the leaders in data science at your own pace, no matter where you are in the world. Each interview is an in-depth conversation, covering the personal stories of these data scientists from their initial experiences that helped them find their own path to a career in data science.

It's not just the early data science leaders who can have a big impact on the field. There is also new talent entering the field, with the opportunity for each and every new member to push the field forward. When I met the authors of this book, they were still college students and aspiring data scientists, full of the same questions that those beginning in data science have. Through 18 months of hard work, they have gone and done the legwork for all those interested, seeking out some of the best data scientists around the country, and asking them for their advice and guidance. This book is the result of that work, containing over 100 hours of collected wisdom with people otherwise inaccessible to talk to (imagine having to compete with President Obama to talk with DJ Patil!). In the meantime, these young authors also have gone on to earn their own stripes as data scientists, working at some well-known companies.

By reading these extended, informal interviews, you will get to sit down with industry trailblazers like DJ Patil, Jonathan Goldman and Pete Skomoroch, who were all part of the core, early LinkedIn data science teams. You will meet with Hilary Mason and Drew Conway, who were instrumental in creating the thriving New York data science community. You will hear advice from the next generation of data science leaders, like Diane Wu and Chris Moody, both former PhDs and Insight Alumni, who are now blazing new trails at MetaMinds and Stitch Fix. You will meet data scientists who are having a big impact in academia, including Bradley Voytek from UCSD and Joe Blitzstein from Harvard. You will meet data scientists in startups like Clare Corthell from Mattermark

and Kunal Punera of Bento Labs, who will share how they use data science and analytics as a core competitive advantage.

The data scientists in the Data Science Handbook, along with dozens of others, have helped create the very industry that is now having such a tremendous impact on the world. Here, in this book, they discuss the mindset that allowed them to create this industry, address misconceptions about the field, share stories of specific challenges and victories, and talk about what skills they look for when building their teams. By reading their stories, hearing how they think and learning about where they see the future of data science going, you will gain the context to think of ways you can both have an impact and perhaps advance the field yourself in the years to come.

Jake Klamka
Founder
Insight Data Science Fellows Program
Insight Data Engineering Fellows Program
Insight Health Data Science Fellows Program

DJ PATIL VP of Product at RelateIQ

The Importance of Taking Chances and Giving Back

DJ Patil is co-coiner of the term 'Data Scientist' and co-author of the Harvard Business Review article: "Data Scientist: Sexiest Job of the 21st Century."

Fascinated by math at an early age, DJ completed a B.A. in Mathematics at University of California, San Diego and a PhD in Applied Mathematics at University of Maryland where he studied nonlinear dynamics, chaos theory, and complexity. Before joining the tech world, he did nearly a decade of research in meteorology, and consulted for the Department of Defense and Department of Energy. During his tech career, DJ has worked at eBay as a Principal Architect and Research Scientist, and at LinkedIn as Head of Data Products, where he co-coined the term "Data Scientist" with Jeff Hammerbacher and built one of the premier data science teams. He is now VP of Product at RelateIQ, a next generation, data-driven customer relationship management (CRM) software. Most recently RelateIQ was acquired by Salesforce.com for its novel data science technology.

In his interview, DJ talks about the importance of taking chances, seeking accelerations in learning, working on teams, rekindling curiosity, and giving back to the community that invests in you.

Since we interviewed him, DJ has gone on to be appointed by President Barack Obama as the first United States Chief Data Scientist.

Something that touched a lot of people from your presentations is your speech on failure. It's surprising to see someone as accomplished as yourself talk about failure. Can you tell us a bit more about that?

Something most people struggle with when starting their career is how they enter the job market correctly. The first role you have places you in a "box" that other people use to infer what skills you have. If you enter as a salesperson you're into sales, if you enter as a media person you're into media, if you enter as a product person you're into products etc. Certain boxes make more sense to transition in or out of than other ones.

The academic box is a tough one because automatically, by definition, you're an academic. The question is: Where do you go from there? How do you jump into a different box? I think we have a challenge that people and organizations like to hire others like

themselves. For example, at Ayasdi (a topological machine learning company) there's a disproportionate amount of mathematicians and a surprising number of topologists.

For most people who come from academia, the first step is that someone has to take a risk on you. Expect that you're going to have to talk to lots and lots of people. It took me 6 months before eBay took a chance on me. Nobody just discovers you at a cafe and says "Hey, by the way you're writing on that piece of napkin, you must be smart!" That's not how it works, you must put yourself in positions where somebody can actually take a risk on you, before they can give you that opportunity.

Nobody just discovers you at a cafe and says "Hey, by the way you're writing on that piece of napkin, you must be smart!" That's not how it works, you must put yourself in positions where somebody can actually take a risk on you, before they can give you that opportunity.

And to do that, you must have failed many times, to the point where some people are not willing to take a risk on you. You don't get your lucky break without seeing a lot of people slamming doors in your face. Also, it's not like the way that you describe yourself is staying the same; your description is changing and evolving every time you talk to someone. You are doing data science in that way. You're iterating on how you are presenting yourself and you're trying to figure out what works.

Finally someone takes a chance on you, but once you've found somebody, the question is how do you set yourself up for success once you get in? I think one of the great things about data science is it's ambiguous enough now, so that a lot of people with extra training fit the mold naturally. People say, "Hey, sure you can be a data scientist! Maybe your coding isn't software engineering quality coding, but your ability to learn about a problem and apply these other tools is fantastic."

Nobody in the company actually knows what these tools are supposed to be, so you get to figure it out. It gives you latitude. The book isn't written yet, so it's really exciting.

What would you suggest as the first step to putting yourself out there and figuring out what one should know? How does one first demonstrate one's value?

It first starts by proving you can do something, that you can make something.

I tell every graduate student to do the following exercise: when I was a grad student I went around to my whole department and said, "I want to be a mathematician. When I say the word mathematician, what does that mean to you? What must every mathematician know?"

I did it, and the answers I got were all different. What the hell was I supposed to do? No one had a clear definition of what a mathematician is! But I thought, there must be some underlying basis. Of course, there's a common denominator that many people came from. I said, okay, there seem to be about three or four different segmentations. The segmentation I thought was the most important was the segmentation that gave you the best optionality to change if it ended up being a bad idea.

As a result of that, I took a lot of differential equations classes, and a bunch of probability classes, even though that wasn't my thing. I audited classes, I knew how to code, I was learning a lot about physics — I did everything I could that was going to translate to something that I could do more broadly.

Many people who come out of academia are very one-dimensional. They haven't proven that they can make anything, all they've proven is that they can study something that nobody (except maybe their advisor and their advisor's past two students) cares about. That's a mistake in my opinion. During that time, you can solve that hard PhD caliber problem AND develop other skills.

For example, aside from your time in the lab, you can be out interacting with people, going to lectures that add value, attending hackathons, learning how to build things. It's the same reason that we don't tell someone, "First, you have to do research and then you learn to give a talk." These things happen together. One amplifies the other.

It first starts by proving you can do something, that you can make something.

So my argument is that people right now don't know how to make things. And once you make it, you must also be able to tell the story, to create a narrative around why you made it.

With that comes the other thing that most academics are not good at. They like to tell you, rather than listen to you, so they don't actually listen to the problem. In academia, the first thing you do is sit at your desk and then close the door. There's no door anywhere in Silicon Valley; you're out on the open floor. These people are very much culture shocked when people tell them, "No you must be working, collaborating, engaging, fighting, debating, rather than hiding behind the desk and the door."

I think that's just lacking in the training, and where academia fails people. They don't get a chance to work in teams; they don't work in groups.

Undergrad education, however is undergoing some radical transformations. We're seeing that shift if you just compare the amount of hackathons, collaboration, team projects

that exist today versus a few years ago. It's really about getting people trained and ready for the work force. The Masters students do some of that as well but the PhDs do not. I think it's because many academics are interested in training replicas of themselves rather than doing what's right for society and giving people the optionality as individuals to make choices.

How does collaboration change from academic graduate programs to working in industry?

People make a mistake by forgetting that data science is a team sport. People might point to people like me or Hammerbacher or Hilary or Peter Norvig and they say, oh look at these people! It's false, it's totally false, there's not one single data scientist that does it all on their own. data science is a team sport, somebody has to bring the data together, somebody has to move it, someone needs to analyse it, someone needs to be there to bounce ideas around.

People make a mistake by forgetting that data science is a team sport.

Jeff couldn't have done this without the rest of the infrastructure team at Facebook, the team he helped put together. There are dozens and dozens of people that I could not have done it without, and that's true for everyone! Because it's a bit like academia, people see data scientists as solo hunters. That's a false representation, largely because of media and the way things get interpreted.

Do you think there's going to be this evolution of people in data science who work for a few years, then take those skills and then apply them to all sorts of different problem domains, like in civics, education and health care?

I think it's the beginning of a trend. I hope it becomes one. Datakind is one of the first examples of that, and so is data science for Social Good. One of the ones that's personally close to my heart is something called Crisis Text Line. It comes out of DoSomething.org — they started this really clever texting campaign as a suicide prevention hotline and the result is we started getting these text messages that were just heart wrenching.

There were calls that said "I've been raped by my father," "I'm going to cut myself," "I'm going to take pills," really just tragic stuff. Most teens nowadays do not interact by voice - calling is tough but texting is easy. The amount of information that is going back and forth between people who need help and people who can provide help through Crisis Text Line is astonishing.

How do we do it? How does it happen? There are some very clever data scientists there who are drawn to working on this because of its mission, which is to help teens in crisis.

There's a bunch of technology that is allowing us to do things that couldn't be done five, six years ago because you'd need this big heavyweight technology that cost a lot of money. Today, you can just spin up your favorite technology stack and get going.

These guys are doing phenomenal work. They are literally saving lives. The sophistication that I see from such a small organization in terms of their dashboards rivals some of the much bigger, well-funded types of places. This is because they're good at it. They have access to the technology, they have the brain power. We have people jumping in who want to help, and we're seeing this as not just a data science thing but as a generational thing where all technologists are willing to help each other as long as it's for a great mission.

Jennifer Aaker just wrote about this in a *New York Times* op-ed piece — that the millennial generation is much more mission driven. What defines happiness for them is the ability to help others. I think that there is a fundamental shift happening. In my generation it's ruled by empathy. In your generation, it's about compassion. The difference between empathy and compassion is big. Empathy is understanding the pain. Compassion is about taking away the pain away from others, it's about solving the problem. That small subtle shift is the difference between a data scientist that can tell you what the graph is doing versus telling you what action you need to do from the insight. That's a force multiplier by definition.

Compassion is also critical for designing beautiful and intuitive products, by solving the pain of the user. Is that how you chose to work in product, as the embodiment of data?

I think the first thing that people don't recognize is that there are a number of people who have started very hard things who also have very deep technical backgrounds.

Take Fry's Electronics for example. John Fry, the founder, is a mathematician. He built a whole castle for one of the mathematical associations out in Morgan Hill, that's how much of patron of the arts he is for them. Then you can look at Reed Hastings of Netflix, he's a mathematician. My father and his generation, all of the old Silicon Valley crew were all hardcore scientists. I think it just goes on to show - you look in these odd places and you see things you would not have guessed.

I think there's two roles that have been interesting to me in companies: the first is you're starting something from scratch and the second is you're in product. Why those two roles? If you start the company you're in product by definition, and if you're in product you're making. It's about physically making something. Then the question is, how do you make? There's a lot of ways and weapons you can use to your advantage. People

say there is market assessment, you can do this detailed market assessment, you can identify a gap in the market right there and hit it.

There's marketing products, where you build something and put a lot of whizbang marketing, and the marketing does phenomenally. There are engineering products which are just wow — you can say this is just so well engineered, this is phenomenal, nobody can understand it, but it's great, pure, raw engineering. There is designing products, creating something beautifully. And then, there's data.

The type of person I like best is the one who has two strong suits in these domains, not just one. Mine, personally, are user experience (UX) and data. Why user experience and data? Most people say you have to be one or the other, and that didn't make sense to me because the best ways to solve data problems are often with UX. Sometimes, you can be very clever with a UX problem by surfacing data in a very unique way.

For example, People You May Know (a viral feature at LinkedIn that connected the social graph between professionals) solved a design problem through data. You would join the site, and it would recommend people to you as you onboard on the website. But People You May Know feels creepy if the results are too good, even it it was just a natural result of an algorithm called triangle closing. They'd ask, "How do you know that? I just met this person!" To fix this, you could say something like "You both know Jake." Then it's obvious. It's a very simplistic design element that fixes the data problem. My belief is that by bringing any two elements together, it's no longer a world of one.

Because of the pace at which the world changes, the only way to prepare yourself is by having that dynamic range.

Another way to say this is, how do you create versatility? How do you make people with dynamic range, which is the ability to be useful in many different contexts? The assumption is our careers are naturally changing at a faster rate than we've ever seen them change before. Look at the pace at which things are being disrupted. It's astonishing. When I first got here eBay was the crazy place to be and now they're on a turnaround. Yahoo went from being the mammoth place to now attempting a turnaround. We've had companies that just totally disappeared.

I see a spectrum of billion dollar companies coming and going. We're seeing something very radical happening. Think about Microsoft. Who wouldn't have killed for a role in Microsoft ten years ago? It was a no brainer. But not anymore.

Because of the pace at which the world changes, the only way to prepare yourself is by

having that dynamic range. I think what we're realizing also is that different things give you different elements of dynamic range. Right now data is one of those because it's so scarce. People are getting the fact that this is happening. It gives a disproportionate advantage to those who are data savvy.

You mentioned earlier that when you were looking to become a mathematician you picked a path that optimized for optionality. As a data scientist, what type of skills should one be building to expand or broaden their versatility?

I think what data gives you is a unique excuse to interact with many different functions of a business. As a result, you tend to be more in the center and that means you get to understand what lots of different functions are, what other people do, how you can interact with them. In other words, you're constantly in the fight rather than being relegated to the bench. So you get a lot of time on the field. That's what changes things.

One of the first things I tell new data scientists when they get into the organization is that they better be the first ones in the building and the last ones out.

The part here I think people often miss is that they don't know how much work this is. Take an example from RelateIQ. I'm in the product role (although they say I'm supposed to be the head of product here, I think of these things as team sports and that we're all in it together), and I work over a hundred hours a week easily. If I had more time I'd go for longer hours. I think one of the things that people don't recognize is how much net time you just have to put in. It doesn't matter how old you are or how good you are, you have to put in your time.

You're not putting in your time because of some mythical ten thousand hours thing (I don't buy that argument at all, I think it's false because it assumes linear serial learning rather than parallelized learning that accelerates). You put in your time because you can learn a lot more about disparate things that fit into the puzzle together. It's like a stew, it only becomes good if it's been simmering for long time.

One of the first things I tell new data scientists when they get into the organization is that they better be the first ones in the building and the last ones out. If that means four hours of sleep, get used to it. It's going to be that way for the first six months, probably a year plus.

That's how you accelerate on the learning curve. Once you get in there, you're in the conversations. You want to be in those conversations where people are suffering at two in the morning. You're worn down. They are worn down. All your emotional barriers come down and now you're really bonding. There's a reason they put Navy Seals through

training hell. They don't put them in hell during their first firefight. You go into a firefight completely unprepared and you die. You make them bond before the firefight so you can rely on each other and increase their probability of survival in the firefight. It's not about bonding during the firefight, it's about bonding before.

That's what I would say about the people you talked to at any of the good data places. They've been working 10x harder than most places, because it is do or die. As a result, they have learned through many iterations. That's what makes them good.

What can you do on a day-to-day basis that can make you a good data scientist?

I don't think we know. I don't think we have enough data on it. I don't think there's enough clarity on what works well and what doesn't work well. I think you can definitely say some things increase the probability of personal success. That's not just about data science, it's about listening hard, being a good team player, picking up trash, making sure balls don't get dropped, taking things off people's plates, being there for the team rather than as an individual, and focusing on delivering value for somebody or something.

If you watch kids running around a track, and the parents want to leave, the kids always answer, "One more! One more!" You watch an adult run laps, and they are thinking, "How many more do I have to do?"

When you do that, you have a customer (could be internal, external, anybody). I think that's what gives you the lift. Besides the usual skills, the other thing that's really important is the ability to make, storytell, and create narratives. Also, never losing the feeling of passion and curiosity.

I think people that go into academia early, go in with passion. You know that moment when you hear a lecture about something, and you're saying, "Wow! That was mind blowing!" That moment on campus when you're saying, "Holy crap, I never saw it coming." Why do we lose that?

Here is a similar analogy. If you watch kids running around a track, and the parents want to leave, the kids always answer, "One more! One more!" You watch an adult run laps, and they are thinking, "How many more do I have to do?" You count down the minutes to the workout, instead of saying, "Wow, that was awesome!"

I feel that once you flip from one to the other you've lost something inherently. You have to really fight hard to fill your day with things that are going to invigorate you on those fronts. One more conversation, one more fight, one more thing. When you find those

environments, that's rare. When you're around people who are constantly inspiring you with tidbits of information, I feel like that's when you're lucky.

Is all learning the same? What value can you bring as a young data scientist to people who have more knowledge than yourself?

There's a difference between knowledge and wisdom. I think that's one of the classic challenges with academia. You can take a high school kid who can build an app better than a person with a doctorate who works in algorithms, and it's because of their knowledge of the app ecosystem. Wisdom also goes the other way: if you're working on a very hard academic problem, you can look at it and say, "That's going to be $O(n^2)$".

I'm a firm believer in the apprentice model

I was very fortunate when I was at eBay, as I happened to get inserted in a team where there was a lot of wisdom. Even though eBay was moving very slowly in things we were doing, I was around a lot of people who had a disproportionate amount of wisdom, so I was the stupidest guy with the least amount of tours of duty. But at the same time, I was able to add value because I saw things in ways that they had never seen. So we had to figure out where that wisdom aligned and where it didn't.

The other side of that was at LinkedIn, when you're on that exponential curve trajectory with a company. People say, "Well you were only at the company for three plus years," but I happened to be there when it grew from couple hundred to a couple thousand people. Being in a place where you see that crazy trajectory is what gives you wisdom, and that's the type of thing that I think compounds massively.

Many young people today are confronted with this problem related to knowledge and wisdom. They have to decide: Do they do what they're deeply passionate about in the field they care most about? Or do they do the route that provides them with the most immediate amount of growth? Do they go compound the knowledge of skills, or do they build wisdom in that domain?

It's a good and classic conundrum. I've gone with it as a non-linear approach: you go where the world takes you. The way I think about it is, wherever you go, make sure you're around the best people in the world.

I'm a firm believer in the apprentice model, I was very fortunate that I got to train with people like James Yorke who coined with the term "chaos theory." I was around Sergey Brin's dad. I was around some really amazing people and their conversations are some of the most critical pieces of input in my life, I think I feel very grateful and fortunate to be

around these people. Being around people like Reid Hoffman, Jeff Weiner is what makes you good and that gives you wisdom.

So for that tradeoff, if you're going to be around somebody that's phenomenal at Google, great! If you're going to be around someone super phenomenal in the education system, great! Just make sure whatever you are doing, you're accelerating massively. The derivative of your momentum better be changing fast in the positive direction. It's all about derivatives.

What do you think about risk taking, and defining oneself?

Everyone needs to chart their own destiny. The only I thing I think is for certain is that as an individual, you get to ask the questions, and by asking the questions and interpreting the answers, you decide the narrative that is appropriate for you. If the narrative is wrong, it's your narrative to change. If you don't like what you're doing, you get to change it.

It may be ugly, maybe hard or painful but the best thing is when you're younger, you get to take crazy swings at bats that you don't get to take later on. I couldn't do half the stuff I was doing before, and I'm very envious of people who get to. And that's a part of life, there's the flip side of when you do have family, or responsibilities, that you're paying for that next generation. Your parents put a lot on the line to try to stay in a town with great schools, and they may not have taken the risk that they would've normally taken to do these things.

If the narrative is wrong, it's your narrative to change. If you don't like what you're doing, you get to change it.

That's part of the angle by which you play. It's also the angle which is the difference between what it means as an individual and team player. Sometimes you can't do the things that you want to do. It's one of the reasons I've become less technical. Take someone like Monica Rogati or Peter Skomoroch, two amazing data scientists and engineers at LinkedIn. What's a better use of my time? Taking a road block out of their way or me spending time debugging or coding something on my own?

In the role I have, in the position and what was expected of me, my job was to remove hurdles from people, my job was to construct the narrative to give other people runway to execute, their job was to execute and they did a hell of a good job at it.

You have talked about your research as a way to give back to the public that invested in you. Is there an aspect of the world that you feel like could really use

the talent and skills of data scientists to improve it for the better?

I think we're starting to see elements of it. The Crisis Text Line is a huge one. That's why I put a lot of my time and energy into that one. But there are so many others: national security, basic education, government, Code for America. I think about our environment, understanding weather, understanding those elements, I would love to see us tackle harder problems there.

Only work on simple things; simple things become hard, hard things become intractable.

It's hard to figure out how you can get involved in these things, they make it intentionally closed off. And that's one of the cool things about data, it is a vehicle to open things up. I fell into working on weather because the data was available and I said to myself, "I can do this!" As a result, you could say I was being a data scientist very early on by downloading all this crazy data and taking over the computers in the department. The data allowed me to become an expert in the weather, not because I spent years studying it, because I was playing around and that gave me the motivation to spend years studying it.

From rekindling curiosity, to exploring data, to exploring available venues, it seems like a common thread in your life is about maximizing your exposure to different opportunities. How do you choose what happens next?

You go where the barrier of entry is low. I don't like working on things where it's hard. My PhD advisor gave me a great lesson — he said only work on simple things; simple things become hard, hard things become intractable.

So work on simple things?

Just simple things.

HILARY MASON Founder at Fast Forward Labs

On Becoming a Successful Data Scientist

Hilary is the Founder of Fast Forward Labs, a machine intelligence research company, and the Data Scientist in Residence at Accel. Previously, she was the Chief Scientist at bitly, where she led a team that studied attention on the internet in realtime, doing a mix of research, exploration, and engineering. She also co-founded HackNY and DataGotham, and is a member of NYCResistor.

What do you do as a data scientist in residence?

I do three things. First, I occasionally help the partners talk through an interesting technology or company. Second, I work with companies in the Accel portfolio. I help them when they run into an interesting or challenging data question. Finally, I help Accel think through what the next generation of data companies might look like.

Do you expect this to be a growing trend, the fact that VC firms are hiring data scientists in residence?

We're at a point where there are very few people who've spent years building data science organizations in a company or building data-driven products. Having people with even just a few years of expertise in doing that is valuable.

I don't expect that this will be nearly as difficult in the future as it is now. Because data science is so new — there are only a few people who have been doing this for a long time. Therefore it really helps a VC firm to have access to someone who they can send to one of their companies when that company has some questions. Right now, the expertise is fairly hard to come by, but it's not impossible. In the coming years, I think more and more people will take this expertise for granted.

What can you tell our readers about the data community in New York City?

We're not a tech city. We are a city of finance, publishing, media, fashion, food and more. It's a city of everything else. We see data in everything here. We have people in New York

doing data work across every domain you can imagine. It's absolutely fascinating.

You'll see people who talk about their work in the Mayor's office, people talking about their academic work, people in health care using data to cure cancer, and people talking about journalism. You can see both startups and big companies all talking about how they use data.

DataGotham is our attempt to highlight this diversity. We started it as a public flag that we planted and said, "*Whatever you do, if you care about data, come here and meet other people who also feel the same way.*" I think we've done a good job with that. The best way to get a sense of New York's data community is to come.

How else do you think data science will change? What will happen to data science in the next five years?

Five years is a long time. If you think back five years, data science barely existed, and it's still evolving rapidly. It will change a lot in these next five. I'm not going to say what is certain to happen in the next five years, but I'll make a few guesses.

We see data in everything here. We have people in New York doing data work across every domain you can imagine. It's absolutely fascinating.

One change is that some of the delightful chaos will go away. I know fantastic data scientists who have degrees in computer science, physics, math, statistics, economics, psychology, political science, journalism and more. People have switched to data science with a passion and an interest. They didn't come from an academic program. That's already changing — you can enroll in Master's degree programs in data science now.

Perhaps some of the creativity that happens when you have people from so many different backgrounds will result in a more rigid understanding of what a data scientist actually is. That's both a good and bad thing.

The second change is, well, let's just say that if I'm still writing Java code in five years I'm going to punch a wall! Our tooling *has* to get a lot better, and it already is starting to. This is a fake prediction because I know things are already happening in this area.

Five years ago, the most interesting data companies were building infrastructure, different kinds of databases. They were working on special tools for managing time series data. Now, the base infrastructure is mature and we're seeing companies that are making it easier to work with those pieces of infrastructure. So you get a great dashboard

and you can plug in your queries, which go behind the scenes and run map-reduce jobs. You won't be spending 40 hours manually parallelizing algorithms and hating your life anymore. I think that will continue to expand.

Culture is also a big part of the practice. I think data culture will continue to grow, even among people who aren't data scientists. This means that within lots of companies, you will begin to see people whose job titles don't say "data scientist," but they will be doing very similar things. They won't need to ask a statistician to count something in a database anymore — they can do it themselves. That's exciting to me. I do believe that data gives people the power to make better decisions, so the more people who have access to it, the better.

How do you think the role of a data scientist will change in a world where every company has data-minded people?

Data scientists will keep asking the questions. It's not always entirely obvious what you should be counting, even for fairly trivial business problems. It's also not entirely obvious how to interpret the results. Data scientists can become the coach, the person who really understands the problem they're trying to solve.

Data scientists and data teams do a variety of things beyond just business intelligence. They also do algorithmic engineering, build new features, collect new data sets, and open up potential futures for the product or business. I don't think data scientists will be out of work anytime soon.

You emphasize communication and storytelling a lot when you talk about data science. Can you elaborate more on this?

A data scientist is someone who sits down with a question and gathers some data to answer it, or someone who starts with a data set and asks questions to learn more about it. They do some math, write some code, do the analysis, and then come to a conclusion. Then what?

They need to take what they've learned and communicate it to people who were not involved in the analytical process. Creating a story that's compelling and exciting for people, while still respecting the truth of the data, is hard to do. This skill gets neglected in many technical programs, as it's taken for granted that if you can do something you can explain it. However, I don't think it's that easy.

Why isn't it easy? Why is explaining something in a simple manner so difficult?

It's hard because it requires a lot of empathy. You have to understand something that's

very technical and complex, then explain it to someone who doesn't come from the same background. You have to know how they think so you can translate it into something they can understand. You also have to do it for people who generally have short attention spans, who are impatient, and who are not ready to spend hours studying.

I do believe that data gives people the power to make better decisions, so the more people who have access to it, the better.

So you need to come up with a solution that uses language or a visualization to facilitate their understanding after you've invested all of this time building a complex model. When you think about it, it's amazing that we can take our complex technical understanding of something and then write it down in such a short, concise way to communicate it to someone who doesn't share the same knowledge or interests. That's amazing.

When you think of it that way, it's not a surprise at all that storytelling is hard. It's like art. You're trying to take a really intense emotion or complex phenomenon and express it in a way that people will understand intuitively.

You've said before that some of the most exciting data science opportunities are in startups. Given your experience with Bitly and advising startups, can you elaborate more on that?

I'll explain with the disclaimer that I'm obviously slightly biased. The most exciting data opportunity is when you have the flexibility to collect data. Often you're collecting data accidentally as a side effect of another product you were trying to build.

Bitly is the classic example of this — short URLs make it easy to share on social networks. You end up collecting this amazing data set about what people are sharing and what people are clicking on across all these social networks. But nobody really set out in the beginning to build the world's greatest URL shortener to discover how popular Kim Kardashian is. Bitly's founder John Borthwick calls this accidental side effect "data exhaust," which is a lovely phrase for it.

That said, if you're in academia, you don't have the benefit of having a product there already collecting data. There's an extra project to do before you even do the work you actually care about. You have to struggle to collect your own data, or go to a company and beg for their data. That's really difficult, because most companies have no incentive to share data at all. In fact, they have a very strong disincentive given privacy liability. So, as an academic, you find yourself in a difficult position unless you're one of those people who are able to build good partnerships (which some people are).

If you're at a larger company, the data you have is probably either stuck in a bunch of incompatible databases or so highly controlled that it will take a huge political effort to get the data into a place where it becomes useful.

Startups are the perfect place where you have a product that's generating it's own data. As a data scientist you have input into how the product changes, so you can ask, "*Can we collect this other thing?*" or "*Do you think if we tried this we might learn something else?*" It's very open as to what you do with it.

I love that aspect that we can learn something interesting from the data. It's a fun process and a good place to be.

What advice would you give our readers who are interested in joining a data science startup? How should one choose where to work at?

Startups are the perfect place where you have a product that's generating it's own data.

Try to learn more about the startup culture. Startups generally have great cultures — one reason is because startups are much more free to have wide variability in those cultures. You'll find that some startups might be a great fit for you, while some of them might feel uncomfortable. There's nothing wrong with you, it's just a company that's not a good match.

This is just good advice in general. When you're looking at working in a small company, make sure it's a group of people that you're comfortable working with and that the social environment is one that you're going to feel happy and comfortable in.

That said, a lot of companies are hiring their first data scientists. Most data scientists have no experience in a job, so it's very hard to find someone who can come in and do a job well that nobody has done before. I would make sure that whoever you're working for — whether it's your COO, CTO or CEO — has a pretty clear understanding of what they want you to do. At least they should be someone you think you could collaborate with in figuring out where you should invest your time.

Can you elaborate more on prioritization and investing time?

You've got an infinite list of questions you can look into — how do you pick the ones that are going to have the biggest impact? How do you do that in an environment where you might have your CEO demanding slides for a board meeting, your head of sales demanding data, etc., and you have a project that you think is really exciting — but no

one else quite gets it yet because they haven't really sat with you and gone into the data?

If you're looking for your first job as a data scientist, I would make sure you have a manager who can manage that process with you. If you're going to be that manager, it's not as easy as it looks from the outside. That is a skill you have to develop. If you're going to be a manager, I'd recommend that you think about those sets of problems -- how to process them and how to communicate them in a way that fits with the process that the rest of the company is using.

What other advice do you have?

Look for good data sets. When I interview people for a data science job, they will already have spent a few hours with people on the team. I'll say, '*You know what we do now. What is the first thing that comes to your mind when you're thinking 'why haven't these guys even thought about this?'*" I don't really care what the answer is, but I want to know that they're capable of thinking about what the data set is and coming up with ideas on their own for what they would like to see.

Most of the answers I've have to that question were things we had already thought of. I don't expect people to come up with genius ideas in the interview, but just to show that they have that creative ability can be really helpful. If you're looking at a company or product to potentially work for and you can't come up with things you would want to work on, that's a problem. You should find something you're a little more excited about.

Do you have more advice on prioritization and making an impact within a company?

During my time at Bitly and in general, we have a series of questions we ask about every data project we work on. The questions would help not just with personal prioritization but also with helping other people in the company understand what was going on.

You've got an infinite list of questions you can look into — how do you pick the ones that are going to have the biggest impact?

The first question is, can we define the question we're interested in? You'd think it would be obvious that it's helpful to write down the question in plain language so that anyone can understand what you're trying to do.

The second question is, how do we know when we've won? What are the error metrics by which we evaluate our solution to this question? If we're working on an algorithm where there are no quantitative error metrics, you at least have to write down that there are none.

The third question is, assuming we can solve this perfectly, what's the first thing we will do with it? I ask that question to ensure that every project is immediately relevant to the business or product. It's not just an irrelevant exercise because we're curious about something. The first thing you'll do with it should also have some longer term implications about what you understand about the data.

For each data project you're working on, you need to ask yourself these questions: what are you working on? How will I know when it's done? What does it impact? If you ask yourself these questions, you always know you're making a good decision about how you're spending your time.

Do you have an example of using these questions to understand a project?

One project you might be working on might be, "*Does our user behavior in Turkey differ from user behavior in the United States?*" That might be an immediately relevant question, maybe because of a sales deal with someone in Turkey.

The longer term goal would be to understand if geography affects user behaviour, and if so, how? You should always be balancing those near-term and long-term rewards, building your library of information of what you know from your data.

The last question is, assuming that everything works perfectly and everyone in the world uses our solution, how does it change human behavior? That question is important because I want to make sure that people are always working on the highest-impact problems.

Another question I ask sometimes is, what is the most evil thing that could be done with this? If I were an evil mad scientist in my volcano lair and I had this technology or knowledge, what could I do with it? You get way more creative ideas for what to actually do with it, very few of which are evil. That's a fun thought experiment to do.

You've given great advice on how data scientists can choose a startup. I wanted to flip that question around — what general advice would you give to new startups that are building their data science team?

This is always a challenge, and often, people have different ideas of what a data scientist coming into the company will do. So this means that first the founders and management team should really understand what they need now.

You're sure that you want some business analytics, product analytics, and metrics. Maybe you have an idea to do something cool with the data — perhaps something that's

well understood like a recommendation engine, or maybe even something that's more creative. But it's hard to find someone who can do all of these things and potentially can grow to manage a team of people.

The things you can do when you're hiring is look for people who learn quickly, are really creative, are flexible, and who can work with your engineering team because that's where they're going to sit. They need to be best friends with whoever is running the infrastructure that holds the data, and they need to be able to work with the product and business side as well.

For each data project you're working on, you need to ask yourself these questions: what are you working on? How will I know when it's done? What does it impact?

That means that you might want to hire somebody who doesn't have 20 years of data experience but who you think can learn really quickly and grow with the product, with the understanding for that person that eventually a team might come around them or they might hire a manager.

So much of hiring well in small companies is finding the right person at the right time for that company. There's no one formula that really describes it — it has to be a good match on both sides.

What advice do you have for students who are choosing between smaller companies and larger companies?

I would say it's worth looking at the smaller companies. The advice I have there is find someone who you'll work for who you think would be a great mentor for a year. Don't just go to a small company because it sounds good. Go to one where you think, "*This is somewhere I can learn from for a year. I think I'll be happy here for about that long.*"

Then after a year, you can re-evaluate. Am I still learning? Am I doing work that I love? And if not, you can move on to your next learning opportunity. But the first few years out of school will help you learn the skills you'll need later. Go to places where you can learn things. That's the best way to think about it.

What other advice do you have for students choosing between companies?

I know when you look at job offers, it's really easy to evaluate them based on how much money you're going to make and where you're going to live. I'm a big fan of living somewhere you like, because otherwise you're miserable all the time, because it's not all about the money. It's most important to be working in an environment where you have

challenging work with people you can learn from.

For example, I once did an internship in AT&T Labs Research, and I loved working there. It was an amazing place full of really amazing people. But I hated living in New Jersey and commuting on the Garden State Parkway. You need to find that right balance of making sure you're in a place where you're going to be happy, but also learning a lot.

Whether you're making 10 or 20 grand more now, versus years later, it doesn't make a difference. As long as you're making enough to have a decent place to live, eat well, enjoy your life when you're not at work, I wouldn't pay too much attention to the salary.

What advice would you give to aspirational data scientists?

A lot of people are afraid to get started because they're afraid they're going to do something stupid and people will make fun of them. Yes, you will do something stupid, but people are actually nicer than you think and the ones who make fun of you don't matter.

My recommendation is that if you're interested in data science, try it! There are a lot of data sets out there. I have a Bitly list of about 100 public research-quality datasets, which you can see here: bitly.com/bundles/hmason/1. You also have access to a bunch of public APIs. You can be creative.

Go to places where you can learn things.

Try to do a project that plays to your strengths. In general, I divide the work of a data scientist into three buckets: Stats, Code, and Storytelling/Visualization. Whichever one of those you're best at, do a project that highlights that strength. Then, do a project using whichever one of those you're worst at. This helps you grow, learn something new, and figure out what you need to learn next. Keep going from there.

This has a bunch of advantages. For one thing, you know what data science is actually like. A lot of data scientists spend their time cleaning data and writing Hadoop scripts. It's not all fun — you should experience that.

Second, it gives you something to show people. You can tell people what cool things you're trying out — people get really excited about that. They're not going to say you tried and you suck, they're going to say, "Wow, you actually did something. That's cool!" This can help you get a job.

A great example of this is my friend Hilary Parker who works at Etsy on their analytics

team. Before she got the job there, she did this fantastic analysis of how Hilary is the most poisoned baby name in U.S. history. The popularity of the name Hilary was growing until Bill Clinton got elected, when it just plummeted. Slowly now it's getting more and more popular again (obviously I love this example because my name is also Hilary). She put it on her blog and ended up getting published in *New York Magazine* — I believe it really helped her land a job by showing that she really knew what she was doing.

I really just encourage people to start putting things up on their blogs and on Github, and not to be discouraged. It takes optimism and stubbornness to do this well.

PETE SKOMOROCH

Principal Data Scientist at Data Wrangling

Software is Eating the World, and It's Excreting Data

Ever since he was young, Pete Skomoroch was interested in science. This led him to double major in mathematics and physics at Brandeis University, where he discovered he enjoyed tinkering with mathematical models and engineering. After graduating, Pete honed his technical skills at Juice Analytics, MIT Lincoln Laboratory and AOL Search.

Pete eventually ended up as a Principal Data Scientist at LinkedIn, where he led teams of Data Scientists focused on Reputation, Inferred Identity and Data Products. He was lead Data Scientist and creator of LinkedIn Skills & Endorsements, one of the fastest growing new products in LinkedIn's history.

He is also the founder of Data Wrangling, which offers consulting services for data mining and predictive analytics.

You're one of the people who've been around data science since the beginning. How have you seen it evolve?

The creation of the data scientist role was originally intended to address some challenges at large social networks. Many software companies at the time had separate teams. There were production engineers, research scientists writing papers and developing prototypes, and data analysts working with offline data warehouses. The classic R&D model required a lot of overhead as ideas were passed from one team to another to be re-implemented. The latency to get an idea into production and iteratively improve it in this way was too high, especially for startups.

The data scientist role was intended to bridge the gap between theory and practice by having scientists who could write code and collaborate with engineering teams to build new product features and systems. At LinkedIn, we wanted to hire scientists and engineers who could develop products and work with large production datasets, not just hand off prototypes. I think the original concept has evolved over the last few years as organizations found it difficult to hire candidates with the full skill set. Simultaneously, as data science became more popular, it evolved into an umbrella term that describes a large number of very different roles. In my case, I was a Research Engineer at AOL

Search and was originally hired as a Research Scientist at LinkedIn before my job title was changed to Data Scientist. In the following years, many business analysts and statisticians also rebranded as data scientists.

Today, depending on the company, a data scientist could be a person who fits that original hybrid scientist-engineer role, or they could be statisticians, business analysts, research scientists, infrastructure engineers, marketers, or data visualization experts. In some organizations, things have come full circle as these skills are held by separate specialized individuals that work together on a data team.

There is nothing wrong with any of these roles and you need all of them for a large modern organization to get the most out of data. That said, I think there is value in having people who fit the original definition, who are interdisciplinary, and can cross boundaries to build new products and platforms. Confusion often arises when companies either don't know which type of role they need for their organization or which type of data scientist they are interviewing.

What [Jeff Hammerbacher] really wanted on his team was a MacGyver of Data Analysis who could work with data, write code in Java and actually implement the algorithms, do some statistics, and really have a good intuition of what would drive strategic objectives.

Can you talk about your story, and how you ended up where you are?

I was really interested in science from an early age. When I started at LinkedIn, I was a research scientist, and before that, I had been a research engineer at AOL Search. The flavor of that role was more like the R&D labs that were doing machine learning research and crunching search query data, but there was a strong pull for us to do more production coding involving product.

I remember a talk that Jeff Hammerbacher gave in which he mentioned that what he really wanted on his team was a MacGyver of Data Analysis who could work with data, write code in Java and actually implement the algorithms, do some statistics, and really have a good intuition of what would drive strategic objectives.

I think that was the kernel of the idea that Data Scientist is a different role. When we are interviewing, we don't want to select for people who are just business analysts who can't code, and we don't want people who are pure engineers who don't have any science or math background. We want people at that intersection. I think that was really the genesis of data science, it is cross-disciplinary.

Some of your undergrad research was about neuroscience, can you tell us a bit more about that?

I was really interested in neuroscience, and physics and electronics. When I went to Brandeis, I found that I actually liked mathematical modeling, data crunching, cracking codes, building models and programming versus doing lots of bio lab work. I felt my real aptitude was digging into the data and coming up with theoretical models, which is what drew me to physics.

When giving advice on undergraduate coursework... [I'd say] take as many physics and math classes as you can, but also learn some computer science.

I graduated college in 2000 while the dotcom boom was still happening. My family was just scraping by financially, so it was really compelling for me to go into industry although I ultimately planned to go back to grad school. I had used Matlab, Mathematica, some C, and Assembly in physics classes and learned Visual Basic in an internship, but I wasn't a strong programmer at that point. In retrospect, that is one thing I would have done differently in undergrad. If I had taken more computer science classes, I probably would have ramped up faster at startups.

When giving advice on undergraduate coursework, I'd echo Yann Lecun, who is now heading AI Research at Facebook and did pioneering work in neural networks. I agree with his advice to take as many physics and math classes as you can, but also learn some computer science.

How did computer science play into your post-college job?

A big piece of what a data scientist is really doing is creating models. It's not just about taking data and loading into a black box machine learning algorithm and running it, but actually modeling something about an organization, a company or a product. It's difficult to find the underlying factors and phenomena that are really predictive and prescriptive vs. something that is just a correlation.

So, when I was looking at jobs coming out of college in 2000, I interviewed at a few places, and one that looked really interesting was a small startup in Kendall Square called Technology Strategy Inc., which eventually rebranded as ProfitLogic, Inc. Our early clients included casinos and some of my coworkers were working on interesting projects optimizing slot machines or spotting cheaters. In the early days we did a lot of consulting work and as it turned out there was a lot of interest from fashion retailers, who wanted things like better inventory allocation and markdown price optimization.

What we were doing was essentially an early version of data science. We would get tapes delivered weekly from big retailers like Macy's or JC Penny or Walmart, and the data would be loaded into our own data warehouses. Then we would run statistical models using a combination of C++ and Python to adjust prices and build predictive sales forecasts at the item level. The ultimate idea was that you could save a lot of time and maximize profit by automatically setting prices using a data driven approach. By taking these optimal price trajectories instead of relying only on intuition, you could make more profit and get more inventory through the system.

My initial role there was similar to a grad student in a research lab. Eventually, I became a hybrid product manager and engineer on the data and algorithm side. I would often be in the office all night, making sure that the weekly model run was working, scrutinizing thousands of charts and logs for model issues. Over time, I started to see areas for improvement and develop my own algorithms for seasonality and other forecast improvements. I was working with people across the engineering teams, the database team and research scientists. That's where I first encountered this pain point of bridging between those areas.

In my case, what I found was that I needed to build up my programming and computer science skills to become more self sufficient. I started out as an analyst building models and then moved into the software engineering organization.

How did you get good at these things? Did you take your own time to learn, or is it more like you just embedded yourself within the groups at the company that you were doing these things at?

I think the only way to excel is to take the extra time. I would go home and read every O'Reilly book I could get my hands on, working through textbooks and side projects. I would do what I could to learn at work, and I was always pushing to work on areas beyond what I was doing before. I'd advise people to take the time to level up early on in their careers, maybe sleep a couple hours less while you can handle it.

The only way to level up was to do real coursework and be around people who were actually doing it.

As I was reading and building models, it seemed like machine learning was a better answer than heuristics or other approaches commonly used in forecast models. I was learning that on my own, but I felt like the only way to level up was to do real coursework and be around people who were actually doing it. There was a job opportunity at MIT's Lincoln Lab working in biodefense, and a big benefit for me was that I could also take graduate courses in that role. I took a fantastic neural networks course with Sebastian

Seung, the author of Connectome, and a machine learning course with Leslie Kaelbling, along with some math courses and an optimization theory course.

My story during that time period is a bit of an unusual one. I would often wake up, go to work in Lexington, go to the MIT library, stay up all night eating from the vending machines and working on problem sets, and go back to work the next day without sleeping. Then I would go home and crash, and then I would repeat that process. I was a zombie for a couple of years and if I could do anything differently, I would balance that much better. Yes, you have to put in your time, but try to balance it. Staying up all night coding is the same thing. Sometimes you maybe have to do it but if you're doing it all the time, you are eventually going to burn out and you are nowhere near as effective as you think you are.

That said, I don't want to make it seem like there is a magic path through this. To get to the point where you can gain the right skills this field does take a lot of hard work and I wouldn't minimize that.

The amount of stuff you have done is unbelievable. I think telling the story of how hard everything was, it's not that you had everything handed to you. That is critical in communicating how people think.

I think there are two parts. Being smart only gets you so far. You have to work hard because anything worth doing is worth doing well and you're better off just digging in. There is this psychological factor of grit that is important.

If you go into management, I advise not giving up coding completely. Own a feature or something that keeps you in the loop.

That is what I would encourage people to think about. Stretch yourself, because if you only work on things that you know well, you're going to plateau. That is part of what makes doing a new startup so appealing. If you go into management, I advise not giving up coding completely. Own a feature or something that keeps you in the loop, so that you're up to speed with the development tools, the build process, the code base, the latest tricks and languages. All these things are important because the further you get from the nuts and bolts, the harder it is to make intelligent decisions. The technology changes rapidly, especially in data science.

Can you talk about your experience at Lincoln Lab? What was it like, especially as you were moving there from the private sector?

There was a mixture of biologists, physicists, hardware engineers and software engineers.

I've always been drawn to the intersection of fields. One project involved a machine learned model for a biosensor. It started as a simple threshold alarm algorithm, and I took it a step further to mathematically model the biochemical processes statistically and apply machine learning on top of that parameterized model.

Anyway, I thought it was interesting that machine learning doesn't just have to be a black box. You can get better results if you have a more intuitive sense or physical sense of what you are modeling and build those features into the model. Often, a custom model is what you need to really nail it. On the other hand, if the answer only has to be 80% accurate, you may want to do something more lightweight.

Afterwards, I moved to DC while my wife was in grad school, but after a few years in defense I wanted to try a job in consumer internet. The most interesting role around DC in terms of machine learning at the time was at AOL Search. The experience working with large datasets at MIT helped me land a role on a great team there mining search query data, and many of my coworkers from that team went on to work at Twitter via the Summize acquisition. There were a lot of management changes at AOL during that time, and I did my best to adapt while things were uncertain, installing an early Hadoop cluster there and experimenting with mapreduce techniques.

There were all these interesting things developing around the same time in the startup world, including the early development of Amazon EC2 and Hadoop, and so I viewed that lack of direction as an opportunity. AOL was very much a content company and I wanted to look at how they could do better in terms of content based on data: Based on search data, what can we decipher about what people are actually interested in, what's trending? And so the first step is to assess, how are you doing versus your competitors? AOL grew through acquisitions, so it wasn't like everything was on a central system. I actually had to crawl internal AOL properties and external sites as well.

Externally, there were signs that data was going to be a big deal, but internally they were dismantling the R&D team, so I knew that wasn't a good place to stay. Another company that I had been talking to in the area was called Juice Analytics. They were primarily known for data visualization, but it was an appealing opportunity to me because I could apply this intersection of skills I'd been developing to product development. So I joined Juice, and we built and shipped a SaaS software product built on Django and EC2. It took about a year, and we were crunching search queries and doing some clustering and pattern recognition to come up with a better picture of your site's search topics instead of just the top ten queries or whatever you got at the time in Google Analytics. That was a great experience of end-to-end product development.

Ultimately, I think it was a failure in terms of product-market fit, but I learned a lot from

that process. As a data scientist in an engineering driven company, you probably go through engineering boot camp, get up to speed with the tech stack, and then you can actually do some engineering to solve your own data problems. When you think about it, that's the way you get leverage in the world that we live in now.

What do you mean when you talk about leverage?

Imagine you have an idea on how to improve your company's product. Say you come in and say, I have this great idea. Everybody will love it and it will make billions of dollars and improve the lives of millions of people. But if you are just describing the idea and you can't implement at least some rough version of it, you are at a disadvantage. That's why I think one of the highest leverage things you can do right now is gain some engineering and computer science skills.

So how did you move from Lincoln Lab to Silicon Valley?

After the experience at ProfitLogic, I was bit by the startup bug and ultimately planned to move out to California. After my wife completed her master's in 2009, we said okay, we're just going out there. The previous year in DC, I became increasingly active on Twitter and I found it really fantastic for finding people with similar interests, especially when you were outside the Bay Area. For data, one of the key people I met was named Mike Driscoll. He's the CEO at Metamarkets, but at the time he had a blog called Dataspora and he did data-related consulting. We contemplated doing an O'Reilly book back then called *Big Data Power Tools* to a) survey these different tools that you should know and b) offer case studies with tips and tricks for practitioners. My vision was that you would hand that book to a new hire and just have them read through it and be ready to hit the ground running. Fast forward to today, and it's really great to see that this is actually happening through a variety of courses, textbooks, meetups and data science bootcamps like the Insight Data Science Fellows program.

One of the highest leverage things you can do right now is gain some engineering and computer science skills.

I think that now a lot of large Fortune 500 companies see the success of consumer internet companies like Google, Facebook, Twitter, Amazon, etc., and they say, "I'm not sure what they are doing, but it seems to be working. I want that. How do I innovate and build products like that?" I think there is a bit of a misconception out there that building dashboards of business metrics like Google will turn you into Google, when really it was a huge amount of engineering infrastructure and algorithmic product development that got them to where they are today. I think a lot of the people who want to get into data

science say, “That is really amazing, how does Google know everything?”.

Or, perhaps “How does Target know I’m pregnant?”

That’s a darker version of that question, but even there it’s interesting to note that the algorithms were really just detecting people following instructions from other software systems. If you are pregnant, there are tons of websites and medical guides that tell you exactly what to purchase and which vitamins to take each week. When you know that, it’s not so surprising that such regimented purchase patterns are detectable.

That said, a lot of data science does seem like magic. How do they create these magical experiences? Even Uber seems like magic (I know that isn’t all necessarily data science), but there is something impressive about getting the cars there fast enough when you push a button that it feels like magic. Fortune 500 companies and big organizations want that magic. And they have some sense that it is happening through data, but they’re not quite sure how. I wasn’t sure either when I started in the field, but it was just clear to me that we were just scratching the surface of what we can do involving engineering and data.

What sort of opportunities did you find at LinkedIn that took advantage of your quantitative background?

The younger a company is, the easier it is to propose new things. When I started working there, LinkedIn had some structured data around titles and companies and company pages, but they didn’t really have any notion of topics or skills. I had just done a bunch of Wikipedia topic mining to build a site called trendingtopics.org, and I thought, with all of these member profiles, I should be able to do some topic mining of the skills that people have. And then I’ll have that structured data set. I thought you should be able to tag people like websites in del.icio.us (which I was a big fan of) and then we would have all this rich data to do better recommendations and matching.

I made a quick proposal to my manager DJ Patil, and I got a time window of six to seven weeks to crank out a prototype. This was back in 2009 and at first, I didn’t think that LinkedIn would have enough data in the connection graph to say how good somebody was at something. But even in early versions, there was a lot of signal in the data and the project was green lighted based on that prototype. At that point, my picture of where this thing was going evolved and I thought that the ultimate value was going to be in the reputation data tied to each skill.

What ultimately led to further enhancements like endorsements was the overarching goal to develop products that fulfilled strategic goals to get people back on the site,

grow engagement, grow profile data, and help improve job matching, ad matching, and other algorithms. The ultimate goal for me was to add a layer of links anchored by skills across profiles, and do for the social and professional graph what Google had done for web pages, allowing people to find and by found.

Can you talk more about what it's like developing new features or products at larger established companies, versus the startups you've worked at in the past?

There was a formal process to bringing new ideas to production at LinkedIn because there may be a big difference between the technologies you used to prototype your idea, and those that LinkedIn is built with. The same thing likely applies for any big tech company at this point. You have to get projects approved and they have to get a budget because you need specialized people on the projects in different organizations: web designers, web developers, frontend engineers, ops people. It takes more of cross-team village to build a product versus a startup where you are a small group wearing a bunch of different hats doing a bunch of different things.

The younger a company is, the easier it is to propose new things.

The spirit at the time during when we built the first version of skills was still that we would try to wear many hats. That said, we wanted to ship product quickly and the way to get that done is to get the right resources lined up so you can really execute. I think one of the worst things you can do is sign up for a project when you know you are not set up for success and you are not resourced properly.

Another important reality to face is that you need to hit product-market fit. You could have a very smart idea as a data scientist, but there is more to succeeding than just having a smart idea. One common problem is that the idea might not align with the company objectives. Another is that many startups that just fail because they are a technology in search of a problem. When you hear there is a shortage of data scientists, I actually believe the most difficult people to find are those that have a more human, intuitive sense of the customer and knack for getting to product-market fit.

How do you develop this "intuition" for product-market fit?

When I interview people, it often manifests itself in somebody who is driven and who has done some novel, creative side projects. When you are building stuff on your own, you often see that your original idea doesn't actually have enough thought put into it. I also like to see when people have worked either in different disciplines or in different areas of domain expertise. An example of a concrete question that would come up in an interview to test for this intuition would be: "If you had access to all of our data, what would you do?"

I think that rather than going from the bottom up and thinking, "What is something cool I can do with this data?" it is sometimes a better approach to think strategically from the top down. "What are the top priorities for this company and what are we going after? What technology or product trends are opening up new opportunities? Who are our customers? What is the market, and how could I do this differently with data?"

This seems to capture the sentiment expressed by Steve Jobs when he said, "People think focus means saying yes to the thing you've got to focus on. But that's not what it means at all. It means saying no to the hundred other good ideas that there are."

Right, and the same thing goes for managing a data team. In the same way, when you're staffing or building a product, think about how well it matches the priorities of the company. LinkedIn could do a million different things, but you want to focus on things that actually align with the strategic goals of the company. There are many things that would align with the vision, but that doesn't mean it's the right thing to do. So you need to prioritize, and you need to do it in the context of all the other things you could possibly do.

It sounds like a great deal of understanding the ins-and-outs of data science is learning how to focus.

Yes. It may not take seven years of focus like a PhD, but you probably need at least a year to do anything of really significant value. If you are coming out of school and you want to work on a data team, you need to find good mentors. You need people who are training you up on the engineering stack, who are sharing the common tools, and helping push projects through management layers. I hear a lot of complaints where lone data scientists feel like they have no support structure. It is really hard to operate without a team on your side, because I think the personality type of scientists is often not the most assertive when dealing with business stakeholders.

Can you talk more about the growing importance of data science within companies?

I think data teams are building really important things. They are actually going about it in a very deliberate way and they're using reason, theory and evidence. People from science backgrounds are well suited for this, because you're building up a theory of what you think will happen if you were to make certain changes to the product. I think that that is really at the core of the skillset that you want in engineering product development and data science to make informed decisions.

I think that data science is going to become this discipline that drives decision-making

and product development. In order for data to have the biggest impact, it needs to be in the early phases of product development rather than just added as an afterthought.

It also involves giving feedback to the product and engineering team about the quality, type and quantity of data that will be collected and affected given certain product decisions. It's incredibly important to have someone sitting in the room and advocating for the data team every time a new product feature is proposed. That may be easier if data science itself rolls up within the engineering or product organization, or has an advocate reporting to the CEO like a Chief Scientist or Chief Data Officer.

If you are coming out of school and you want to work on a data team, you need to find good mentors.

Of the people we have talked to, you can offer a unique perspective on how to effectively manage a data science team because you are so engineering focused as well. There are a lot of managers who are very people focused or they sort of try to massage the politics of the company to get things done, but you seem to want to stick very closely to the nuts and bolts of a company. So what do you find to be effective in creating a data science team?

Jeff Weiner had this framework for prioritizing decisions around vision, strategy, mission and objectives. He used it as a leadership framework and a way to rally people behind a vision. Of the things that I think an effective engineering manager needs to have, one of them is expertise. If you don't understand what the people on your team are doing, you're going to have a hard time making the right calls. Beyond that, you need to be an advocate for what is right for the company and by proxy what's right for your team.

A good leader for a data science team understands some data science, has some vision to see what the right path is, brings the right people in, gets the resources, and then gets out of the way and gets other people out of their way. If your team is being thrashed around and pulled in different directions, it will be hard to stay focused.

There's a great talk by an MIT professor named Fred Kofman who has a book on business strategy called *Conscious Business*. He says when most people are asked what their job is, they reply with their job title. But that's a very limited way to think about the role you play within a team or company. If you use the analogy of a soccer team, the various players all may have different roles. For example, as the goalie your job is to simply stop the ball. As an attacker your goal is to score the most number of points. But if you are completely optimizing for these local metrics, your team still might not win! So I think that what can really make teams successful is everyone really believes in what they're doing, believes in the mission and feels like they're enabled to accomplish that.

How did your perspective change throughout your own life?

Earlier in my career, I thought what was blocking me from more success was not having the engineering skills. Over time, as LinkedIn went from 300 to 5000 employees, what's often difficult for organizations at that scale is communication and coordination issues. What I often would see blocking people was of that nature. It was less pure engineering ability, and more: "How do you get stuff to ship? How do you get resources? How do you get priority?" If I were given total freedom, I would actually just enjoy building stuff and building algorithms, but when you want to maximize impact and success at the company, I think more of what was blocking me at that stage was having to navigate structure within the company.

He says when most people are asked what their job is, they reply with their job title. But that's a very limited way to think about the role you play within a team or company.

My two cents of advice on that would be engineering, engineering, engineering. Because in that environment, or at Facebook or Google, optimizing and getting that right is really going to enable you much more than other alternatives. In a larger company, you're always going to have challenges of organization and your throughput isn't going to be as high by definition. That's why startups exist.

I really like what you said about the fact that even if you wanted to build, build, build, it would be more effective for a team to unblock their processes. That tied in very closely with the analogy you have with soccer players. Some soccer players want the personal glory but the best soccer players are the one who realize it's the team winning that is more important than fulfilling their own goals.

I think it's a balancing act, and I would add one other thing. My opinion has been the best way to show the way is just to do good work. But it's not enough to just do good work; you also have to talk about it. That's something else you can pull from science because a big part of science is communicating. There's value in that, both from a recruiting respective, but also for training the next generation. I think that's the way we build and weave on top of each other's experiences, it's all connected. I would balance talking about your projects with building. I would say work hard, work for a long time, and then talk about what you did and go on to the next step.

Given all of your experience and perspective, what do you think is going to be the future of how data is used in the world?

Four to five years ago, I think investors were a good proxy for the future. They might not

come up with all the ideas, but they hear a lot about what people think and are cued into where things are headed. It was very early for the data space and people were building low-level backend technologies. Over time, interest started to shift to what gets built on top of these backend technologies.

I think what people are really thinking about now is how to replicate the Google and Netflix approach and map it onto the rest of the world. There is a much bigger wave coming of building tools and applications on top of all of this data and infrastructure. There now exist data companies in oil & gas, and health care and other areas, taking on sorts of different verticals.

I'm looking forward to seeing a set of data companies like this. I think all the data companies that are building better platforms and better tools are making everyone's life easier and I want to see more of that, but I also want to see more industries disrupted in a way where it makes society more efficient and people's lives better.

I think the other wave we're hitting is that of social data. All the social data that is being generated is really instrumenting the world and people's behaviors in an entirely new way. Everyone has a Facebook account, a LinkedIn account or a Twitter account, which provides immediate context about the person. We're never lived in a time where there's so much context about you readily available to make your daily experience better. The other key component of this is that we all have mobile computers in our pockets, generating all this ambient data.

We're going to see more smart software at the intersection of those two trends. For example, why does it take four hours to book a flight right now? There are all these workflows, suboptimal systems, and paperwork which could be much easier using mobile and social data. In movies like *Her*, you are starting to see where the rise of Google Now, Siri and things like that could be headed. One thing I think is really interesting is this entire field of intelligent systems. That's a common thread in things I've worked on.

I think that having these techniques and this intelligence in that sea of data acting on your behalf is the next stage. You have context, you have alerting, you have all these disaggregated unbundled verticals like Pandora, but I think next you're going to see this really cool future where you're going to express a desire and intent and something else is going to make it happen. I think that's what I'm most excited about, and why I think for data scientists, the world is your oyster.

MIKE DEWAR

Data Scientist at *The New York Times* R&D Lab

Data Science in Journalism

Mike Dewar is a Data Scientist at the New York Times R&D Lab. Mike holds a PhD from the University of Sheffield, UK, where he studied the modelling of complex systems using data. His current work now focuses on building tools to study behaviour.

Before joining The New York Times, Mike worked at the New York tech company bit.ly, and completed postdoctoral positions at Sheffield, Edinburgh and Columbia Universities. In this interview, you'll read Mike's stories about fruit fly necrophilia, how The New York Times looks into the future and ways that data science is affecting journalism.

Mike is a data ambassador for the non-profit organization DataKind, and has published widely on signal processing, machine learning and data visualization.

Can you trace your career path for our readers? What got you interested in data science? What got you interested in bitly and The New York Times, and what projects have you done that you can share with our readers?

I got my PhD in Modelling Complex Systems from the University of Sheffield in the UK. The department is called Automatic Control and Systems Engineering, which in the US is sometimes called Controls or Cybernetics — it's the study of feedback, modelling, and control.

My PhD looked at modelling spatial-temporal systems. The idea is that you would collect data from the physical space and then build dynamic models of how the system evolved through time using the data you collected.

Then I did a few postdoctoral positions. I did a post-doc at the University of Sheffield for a year. We worked with Unilever and I looked at modelling how people were brushing their teeth. By attaching sensors to a toothbrush with accelerometers and positional sensing, they collected all this data about how people brushed their teeth — it was a very strange gig.

I did that for a year, spent some time writing up the papers for the PhD, and then I decamped to Edinburgh University, where I worked in the School of Informatics, studying

the behaviour of fruit flies. The biologists would alter the brain of the fruit fly and observe their changes in behaviour. In courtship behavior, specifically, the changes were easy to see. If you place a male fruit fly in a small space with a female fruit fly, even the dead body of a female fruit fly, it will mate with her. Well, it will definitely try at least, which is a bit grim.

So, there were loads of fun modelling of sequences and some nice machine learning. I even got to learn how to prepare mutant fruit flies. Most of this work was done at Edinburgh but also included a little bit of work at Harvard, at the Longwood Campus. Then I got the gig at Columbia, which was in the Applied Physics and Applied Math Department. That was with Professor Chris Wiggins, who you might have come across in your studies of data science.

Essentially, leaving academia is a moment where you have to decide if you want to be a professor or not, and I think I'd already decided that was not quite what I wanted to do.

He and Hilary Mason wrote a blog post which outlined various steps of data science, namely: "Obtain, scrub, explore, model, interpret." The steps outlined this idea of a data science flow being practical and producing tangible outputs. Chris was thinking a lot about that with Hilary while I was studying T-cells.

There are lots of different types of T-cells - the population of these different T-cells in your body changes before, during and after an infection (this is how immunization works). So after an infection, you have "memory" T-cells in your body. The group at Columbia was very interested in how T-cells change to this "memory" state.

They collected lots of genetic data and looked for different genes that were responsible for changing the state of these populations of cells. You'd be working with 8 microarrays, but each microarray would have 25,000 genes on it. You had a very strange machine learning problem, whereby you had very little data to go on but it felt like you had a lot because of all the features.

It was through Chris Wiggins that I met Hilary Mason, who was my boss at Bitly. I had also become engaged to a girl who lives in New York, so when it came time to start thinking about what was next after my post doctorate at Columbia, it was important to me to stay in New York. But life as a postdoctoral student in New York sucks because it's quite expensive here. At the same time, the idea of "big data" was just coming to the forefront. There were numerous social media companies that were just starting to think about what they might do with all their data. I was interested in behaviour and making tools for studying behaviour, so Hilary showed up at just the right moment when I wanted to

pay the rent, stay in New York, study behaviour, and use lots of data while doing it.

So I jumped ship and went to Bitly as a data scientist. I think I'm probably amongst the first people who had that title. I made tools at Bitly for studying very large numbers of people's behaviour and trying to build interesting, potentially profitable streams.

Bitly ran its course. I was there for about a year and a half. We did lots of interesting things, but it became time to move on. About that time, a position at The New York Times R&D Lab showed up, which was somewhere I'd wanted to work for years, so I moved over to the lab where I've been now for about two years doing all sorts of interesting things.

Essentially, leaving academia is a moment where you have to decide if you want to be a professor or not, and I think I'd already decided that was not quite what I wanted to do. I like coding and making things, but I don't enjoy talking all day, so that was the decision I made.

We've been talking to a lot of people who decided to jump ship from academia. It seems like a lot of them have been citing reasons such as the lack of dynamism. They felt that data science was much more interesting and fast paced. Did you feel that as well?

No, not really. Academia was very fast paced and very intense, with cutting edge research. The stuff I got to work on was amazing. Watching the very modern imaging of T-cells changing and learning about viruses was overwhelmingly fascinating. When the practical wasn't going quickly, the theoretical was going quickly, and there were always ten different things to do.

I had a very interesting time in academia. Postdoctoral positions are great fun. Lecturing, however, didn't look like so much fun. I wanted to hold onto the fun bits of academia and get paid, which is no small thing when you are starting a family. When staying in New York, which is a bizarrely expensive place, a certain set of constraints comes your way. In short, academia was amazing.

It seems like from your academic background you learned a lot from looking at a complex system, a mass of data, and extracting stories and hypotheses from that. You talk about how the unifying theme in data science is actually just identifying massive behavioural phenomena. What is your advice for identifying the questions, telling the stories, and identifying the hypotheses in the data set; especially since you say data science is all about abductive reasoning, finding stories, and learning from data? What is your advice for learning which story to tell with data and what to look at?

The key piece of advice is always to draw lots of pictures and draw them very quickly. Draw pictures of how things work, even just flow diagrams or engineering block diagrams. Make very rough, quick visualizations of what's in the data, starting with time series and histograms. Thinking hard about graphical modelling and really trying to get to grips with the system and data set that's in front of you helps you think about how the probabilities fit together.

Make very rough, quick visualizations of what's in the data, starting with time series and histograms.

The danger that I see people getting into is that the drawing of the picture becomes the last thing you do, like when you're reading an academic paper. The results and pictures are always at the end of an academic paper, which is a terrible shame. I think the paper should start with pictures of time series and distributions, and go from there into the theory. That's often how we work.

That would be my very general advice: to fail early and to fail often. It's okay to draw lots and lots of pictures that might all be rubbish, but if you draw pictures quickly and really start to understand what's actually going on, you begin to get much deeper ideas of what the right questions are, than if you just start with a classifier.

Can you elaborate on drawing pictures a bit more?

I learned a lot at Edinburgh about graphical modelling, which is a very simple technique for exploring conditional probabilities and trying to explore how random variables in a system affect one another. The beautiful thing about graphical models is that if you start drawing them, you are, at the very same time, beginning to explain your assumptions about the system. Also, you're starting to do quite a mathematical task of imposing some structure that you can then test. I really enjoy quickly trying to show whoever I'm working with a graphical model of how I think things work. The conversation gets going very quickly and it leads to testable hypotheses, which is great.

The other interpretation of drawing things quickly is to get immediately into the data set. As soon as someone hands you a data set or gives you access to a stream, the very first thing to do is to find an interesting variable in the data set and plot it. If it's over time, plot a time series. If you've got lots of samples of that variable then plot a distribution. If it's both then plot both. You can do that using Python, or R, or Tableau, or Excel. Do that first and don't waste time. It takes five minutes to make some plots.

The reason is that it gets you thinking about your assumptions in the same way that graphical models do. The distributions and the time series get you thinking about the

data. Both of those together are the beginning of a modelling process that will see you in good stead. It's quite an iterative process. If all you've got is a Bash terminal, then I would sort my data and then pipe that to "uniq –c" to get a really cheap histogram.

You say that visualization and communicating data is very important because it helps other people generate hypotheses and trust the data. What advice do you have for the best way to approach making visualizations to an internal company audience?

One thing we've been doing lately is trying hard to show all the data. I would normally start by thinking about how a system is working and what I'm trying to get out of a data set. Then I would draw some aggregate visualisations, for example, a histogram if I'm interested in how things are distributed, or a line plot if I'm interested in a time series.

As soon as someone hands you a data set or gives you access to a stream, the very first thing to do is to find an interesting variable in the data set and plot it.

One thing we've tried to do more recently is to draw every single data point in a visualisation, rather than aggregating, just like in a scatter plot. This is something made much easier as I now get to work regularly with Nik Hanselmann, who is a creative technologist in the lab and is extremely adept at this sort of thing. If you can make a scatter plot of a large data set interpretable, then that act of showing all of the data points allows people to see a zoomed out view of the whole thing and allows them to pick on individual data points. They see the outliers and wonder why there's an outlier there.

Clusters are another good example. If you've done your scatter plot well and people can start to pick out different features of the scatter plot by looking directly at the bit of data that you want to show them, then they start to ask questions and start to wonder. That helps you as the analyst or the data scientist. It helps you in trying to understand what your audience is actually interested in and how you might help them make decisions. It's an incredibly difficult thing to do without some sort of interaction like that. Trying to show all the data points is quite challenging sometimes, but that's been oddly effective over the last year or so.

Other than that, axis labels. I feel old saying it but lots of people don't put axis labels on things. You read lots of blog posts about lying with statistics and all that sort of stuff and all the tricks people play, which is fine, but it's very difficult to get away with those tricks if you label your axes properly. You shouldn't trust any graph that doesn't have their axis labelled properly.

How do you think this whole explosion of data, as well as computational power and analysis on top of that, is going to affect the nature of journalism?

The reason that I ask this question is that where I went to school at UC Berkeley, there were actually quite a few workshops held for students who wanted to go into journalism, but these workshops weren't at all about journalism. They were all about D3, Javascript, Python, and R. To somebody who doesn't have as much background knowledge, how would you describe what's going on regarding big data and journalism?

There are a few parts to your question. There have been computer-assisted reporting (CAR) journalists for a long time now. Our computer-assisted reporting desk has been around for many years so the idea that data has been affecting journalism is not a new one.

This is how I came to grips with the term "big data." A friend of mine pointed out that we should think about big data like we think about punk — a cultural moment that was meaningfully hyped for a period, which then led to a lasting change in society.

I like this idea of "big data" as a cultural moment because there's been a definite change in the amount of data we can collect and the expectation of collecting data in the first place. The standard costs of storage, processing, and transmission have all gone down. There has been, over the last few years, a dramatic cultural shift in and around data storage. That hasn't gone by journalists — it's quite the opposite. What they're faced with are huge data sets that they think might contain stories. Or it'll be the other way round where they believe that there is a story and will use the FOIA (Freedom of Information Act) to get data sets.

A journalist will often assume that there is data associated with a story and will demand to see it.

When they're telling a story, or they believe that there's data associated with the story, they will search government organizations or FOIA organizations that have worked with the government and are subject to the Freedom of Information Act. I think the rise of the FOIA is an interesting response to big data in the sense that a journalist will often assume that there is data associated with a story and will demand to see it.

Rather than the WikiLeaks style, where there is a huge data dump for one reason or another, journalists will believe that there is data associated with their story and will use FOIA data in order to support the story. This is a lot more work, and that work is a lot more laborious. The impact of big data is a culture where we expect there to be data.

The other side of that, which I think is probably a bit sexier, is the WikiLeaks side of things, where there is a huge pile of data that is being made available - like the Medicare data. There's a huge data set that's been released around how Medicare dollars are spent with personal information about doctors that receive Medicare funding. There have been a lot of stories out of that data set that are very interesting. That's the other mode that journalism works in. That's when people want to use R, or Python, to clean and analyse the data and d3, ggplot, or matplotlib to build visualizations of that data set. D3 is especially interesting because it's used to make web and print graphics, which is why you see it a lot.

What can you share with us about what you do at the R&D Lab, especially since most of the people we've been talking to work at technology companies, instead of a journalism company with a very strong technology component?

The R&D Lab was set up in 2006 to fulfill a number of roles. Specifically, it tries to think three to five years into the future, tracking social, cultural and technological trends relevant to The NYT. That gives us quite a range of possible projects.

We're thinking a lot lately about how to extract information from article data.

The other function of the R&D Lab is essentially to listen. That takes two forms. One is a futurist approach where we try and watch what's going on in the blogosphere and watch what's going on in new technologies. We try and keep an ear out for anything that looks like it might have something to do with the future.

We also act as a gateway. If someone is developing a new, interesting business software that they think The New York Times might be interested in, but there's not an immediately clear use case, often we'll speak to them. We'll ask them some questions and try and understand what they think the future looks like. We can think about how that fits in with how The New York Times thinks about the future.

In terms of projects, it's quite varied. We're thinking a lot lately about how to extract information from article data. Given an article, can you extract all of the statistics, the quotes, facts and events? This is a tired old problem, so we're trying to think about other ways we might accomplish that. Can we capture information like that during the writing process or the editing process or the production process, rather than approaching the articles with an extractive, natural language processing view? It would be much more interesting to see what metadata we could generate in the first place.

That's an example of journalistic stuff. Then, we think about how the news might be

presented in the future. One example is a good idea that the lab had regarding the future of tablets. The New York Times R&D Lab had thought about what a tablet reader application would be like well before the iPad came out. When the iPad did come out, The New York Times had a head start in understanding how people might interact with their tablet and what would be interesting to show on it.

What advice do you have for other PhD students and people in academia transitioning to data science, especially since you've been through this already? What advice would you give someone interested in transitioning to data science?

Code in public.

Code in public, that's number one. If you're going to be a data scientist, you're probably going to have to be able to program in one way or another. There are lots of different options, but you're probably going to have to be quantitative and be able to write non-trivial programs on the computer. As you code, as you practice, as you go to hackathons, as you code for your post doctorate or for your PhD or for your graduate degree, make sure you do it in public. Put it on Github. To a certain extent I'm on the other side of it now where I put every thing I think of on Github, so it's a bit of a mess.

Especially with PhDs, one of the problems we see is that although they come from impressive universities, they have impressive resumes, and they've written these nice papers, but we still have no idea if they can actually write code. That makes them more difficult to hire.

Coding in public also encourages you to engage with communities that you work with. There are programming communities that share your languages; academic communities that might want to use your code to test out your claims; and companies that want to evaluate you and reduce the risk in hiring you.

The other thing is networking. It's more or less the same thing, but it's important. In major cities it's very easy for you to get out of your office or house and visit meetups and user groups to give a talk. Giving talks about your academic work to lay people is an incredibly interesting and enlightening experience, one that you should go through. It also exposes you to the business communities and the various kinds of people that you might want to get jobs from in the future. It also shows you what other people are up to; it knocks your academic naivety very quickly, which is great.

Other than coding in public and networking, try to apply all your work to something. I wrote three papers for my PhD, and they were all about the EM algorithm. There's a load of spatial-temporal models that I put a lot of work in to. Over 3-4 years, I wrote

some papers, and nobody cared, nobody at all. However, when we applied this theory to modelling troop movements in Afghanistan, lots of people cared. We won awards. We wrote a book. We were in the news. The idea of taking the advanced things that you learn at school and applying them to something important and meaningful exposes you to a world that's difficult to see from the incremental science of being a good student.

RILEY NEWMAN Head of Data at Airbnb

Data is the Voice of Your Customer

Riley Newman paid his way through college at the University of Washington by being part of the US Coast Guard. After graduating with degrees in economics and international studies, Riley pursued graduate studies in the UK at the University of Cambridge, before he was called back to the US by the Coast Guard.

After working for a few years in economics consulting, Riley met the founders of Airbnb, and was drawn to their vision and focus on culture. He ended up joining Airbnb as one of the early employees.

Now, Riley is the Head of Data Science for Airbnb where he data science teams using data data to listen to customers' voices and desires.

Can you explain a bit about your background and how you came to Airbnb?

I went to college in Seattle where I majored in international politics and economics. Halfway through my time there, I realized the value of statistics for understanding social trends better. This was something I deepened in grad school, and wanted to pursue in a PhD, but I had joined the Coast Guard in undergrad to pay for school and they called me back from the UK after my master's. So I came home to the Bay Area with the plan to get some experience working with data while completing my Coast Guard obligation and then planned to head back to the UK for the PhD.

I spent three intensive years working with a group of economists that were modeling the 2008 recession. One of them had a degree in computer science and taught me the value of automating analytical processes, which I found intriguing. When the time came to leave for the PhD, I was torn — I only wanted to do it for the technical training; my heart wasn't in academia, and I was a bit tired of consulting. Serendipitously, I met the founders of Airbnb through a mutual friend, right as I was struggling with this.

There are a couple of things about Airbnb that resonated with me. First and foremost, the concept behind the company. In undergrad, I read a lot about globalization and the growing interconnectedness of the world; also about the fundamental sustainability issues associated with this trend. Airbnb struck me as a solution — it would facilitate more travel and bring international communities together without requiring the construction of additional structures.

I was also attracted to the founders' focus on culture. This is something I hadn't experienced in previous roles. They placed so much value on the sense of camaraderie on the team — more so than high school and college lacrosse teams, Coast Guard units, or the consulting firm — and the impact that brings to our work. Looking back, I think this is the "secret sauce" of Airbnb's success.

Finally, I was excited about helping to build something. As a consultant, I had exposure to a wide variety of problems but, at best, we could convince the client that our work was actionable. At Airbnb, I would be able to follow the analysis all the way through to impact. And startups are fast-paced environments where you can see the impact of your work on a daily basis. That was really exciting to me.

They placed so much value on the sense of camaraderie on the team. Looking back, I think this is the "secret sauce" of Airbnb's success.

How does this tie into the industry buzz around "Big Data"?

"Big Data" is such a common term these days. I heard a joke recently, "What do big data and teenage sex have in common? — Everybody is talking about it. Nobody knows what it is. All their friends say that they do it, so they say that they do it too."

Like all buzzwords, Big Data is getting tiring. But I met with a more seasoned data scientist recently who described the field in the '80s and '90s — there was much less data so they needed to use advanced statistical methods to identify simple trends. These days, with the volumes of activity web companies are able to generate, and the depth of storage facilitated by technologies like Hadoop, we're able to gather and make use of much more data. So it's more of a question about how to sift through it all. I think this has made computer science degrees that much more valuable.

I have a data scientist friend whose resume begins with three things he firmly believes: more data beats better models; better data beats more data; and the 80/20 rule. I couldn't agree more.

I think that is a really good introduction as to what you think data science is. I want to go back to something that you mentioned earlier; your Master's degree was in Economics, is that right?

Yes, I was in the applied economics department at Cambridge. My research was in the field of economic geography/spatial econometrics.

Many people we've interviewed have their PhD or Masters in physics, statistics, math, or computer science. You're one of the few data scientists we've talked to with an economics background. Do most people on your team tend to come from backgrounds in the hard sciences or are the social sciences represented as well?

More data beats better models; better data beats more data; and the 80/20 rule.

Everyone on the team has some degree of quantitative training but I like having a wide variety of backgrounds because this brings different skill sets and approaches to solving problems. For example, computer scientists are great at scripting automated solutions and productionizing models; statisticians ensure our models are rigorous; physicists are very detail-oriented; and economists can build frameworks for understanding problems. Airbnb is particularly interesting to economists because of our two-sided marketplace, which lends itself to modeling supply and demand, and looking for ways to make our markets more efficient.

But the key thing is that everyone on the team is able to drive impact in the company through the cultivation of insights drawn from data. I'm less interested in what people studied in undergrad than in their ability to do this successfully. However, this requires a solid grasp of statistics, experience in coding, and great communication and problem-solving. Our interview process exposes these skills very well, so we're able to consider people from less traditional backgrounds.

I agree with you that there isn't necessarily one particular field that data scientists come from. However, it does seem like most people come from certain fields that tend to teach some of the most relevant skills. Building on that, what would you say are some of the most valuable or relevant skills that someone in academia should build right now?

Many people coming into data science from academia have honed their ability to think mathematically or statistically and, to some extent, work with data. The big division that I see is the ability to lend those skills towards problems that will result in an actionable solution. In other words, the types of questions they ask are as important, or more, than the methodology behind solving them. In their research, they focus on why something is the way it is or how it works; in industry we're more interested in what we should do. If the how or why lends itself to answering this, great. But if nothing changes as a result of your work, then it wasn't that valuable.

When we ask other data scientists that question, we hear about technical skills like Python and programming. We don't hear as much about extracting actionable insights.

I'm not saying that those aren't relevant; I'm presupposing that anyone hoping to generate actionable insights from data has the ability to work with the tools of the trade. At Airbnb, we mostly use Hive, R, Python, and Excel.

When we ask other data scientists that question, we hear about technical skills like Python and programming. We don't hear as much about extracting actionable insights.

When we interview people, our process is very transparent (see Quora post on this, here). We give candidates a day to solve a problem similar to something we've faced, using real (but anonymized) data. They spend the day seated with the team and are treated like anyone else, meaning they can collaborate with anyone. At the end of the day, we have them walk us through what they found and tell us what we should be doing differently as a result. This is too tight of a timeframe for someone to learn a tool while trying to use it to solve the problem. Their time needs to be completely focused on getting to that actionable insight.

Continuing along that vein, you've been talking a lot about data scientists coming from a Master's or a PhD. I'm also wondering about your opinion of people coming in with a Bachelor's in a quantitative field or similar?

People can absolutely break in with just a Bachelor's. We shifted to the interview model I described earlier because we realized our image of a data scientist was yielding false negatives. If you have the right mindset, a decent understanding of statistics, and can use SQL and R, you'll be able to get a job.

This is particularly true in younger startups. When I think back to the early days of Airbnb, we were able to squeeze a lot of growth out of a simple ratio. If I spent a month building a perfect model, I would have wasted 29 days. As a company matures, so does (hopefully) its understanding of its ecosystem. So there's a need for more sophisticated approaches.

You started at Airbnb when it was in a really early stage. Now you're at the point where it's growing very fast — it's become a large company. What are some of the ways you've seen that transitioning into the work that you actually do?

I see this transition shaping our work in two ways. First, the team is big enough now that we're able to go much deeper into problems. In the past, we were jumping from one fire to the next, so we weren't able to invest large amounts of time into a single problem. And that's natural for a startup. But as the team has grown, we've been able to focus on some of the key topics for the business and understand them more deeply. We also now

have people on the team building data products, which is exciting.

Second is the democratization of information. We're not the only team that has grown over the last few years and everyone is hungry for data to guide their work. So we have to find ways to remove ourselves from the process of answering basic questions. The last thing you want to be is the gatekeeper of information because you'll spend all of your time responding to ad hoc requests. So we've invested a lot in the structure of our data warehouse and the tools used for accessing it so that it's intuitive to people with less experience working with data.

What do you think are some of the most fundamental ways in which data science can add value to the company?

I think data can add value everywhere. It's the voice of your customer — data is effectively a record of an action someone in your community performed, which represents a decision they made about what to do (or not do) with your product. Data scientists can translate those decisions to stories that others can understand.

When I think back to the early days of Airbnb, we were able to squeeze a lot of growth out of a simple ratio. If I spent a month building a perfect model, I would have wasted 29 days.

We spend a lot of time with our product team, which is the most traditional place for a data scientist. There's a wide range of work happening here. For example, our trust and safety team builds machine learning models to predict risk and fraud before it takes place. They also have to think about ways to measure intangible things, like the strength of trust between people in our community so we can identify ways to improve this.

We have other people working on matching guests and hosts, improving the model behind our search algorithm and uncovering new features to improve the match. A while back we published a blog post about this here.

With our mobile team, we try to uncover opportunities for improving the app. One guy on the team looked at the probability of performing an action on the app relative to how far away that feature is from the homepage. This obviously showed that the more buried something is, the less likely it is to happen — but it's a framework the mobile team can now use to think through the structure of the app.

But we don't just work with product. We think about user lifetime value and growth opportunities with our marketing team, operational efficiency with our customer support team, and we've even been chatting with our HR team about how they can leverage data

to better understand recruiting and career growth.

I try not to segment our work by stakeholder; rather I look at the key drivers of the business and try to figure out what problems need to be solved in order for Airbnb to be better, and then figure out who is in the best position to use that information.

So we've invested a lot in the structure of our data warehouse and the tools used for accessing it so that it's intuitive to people with less experience working with data.

Airbnb is a transactional business so there's a funnel we can break apart and analyze. And getting back to the concept of data being the voice of the customer, we always start by looking to our community for advice on what to do next. At the top of the funnel we try to understand how people are hearing about Airbnb. We can use online and offline marketing to drive this, emphasizing growth where we think there's a strategic opportunity or where we're seeing positive ROI (return on investment). For this, we begin by looking to our community for ideas; for example, where many people are searching for places to stay but we don't have enough supply to accommodate them. If this isn't an anomaly (e.g. one-off event), it represents an opportunity for growth.

Next is the experience people have when they come to our site. There's a lot of A/B testing here, looking for ways to make it more intuitive and satisfying to a person of any demographic, anywhere in the world.

After that is the offline experience, which is tricky because the data behind this isn't as rich as site usage. But we can get a lot from the reviews people leave each other - a combination of quantifiable ratings and NLP we can perform on the text of the review.

Finally, we look at what we can do to get people to come back and try it again. Mostly, this means improving each of the steps above, but we think about experiences people have with customer support or community groups as a way of staying connected.

The final thing I would love to get your perspective on is just looking towards the future. Where do you think the future of data science is and where do you think we are relative to what data science could be in the future?

I think we'll see a lot of growth on the tools front. It's amazing how quickly Hadoop and Hive have matured just over the last few years and there are new and exciting technologies emerging almost daily. So I'm hopeful that we'll eventually have lightning-fast tools that can work with data of any size.

I also think data logging will develop a lot, because people are aware that you only focus

on what you can measure, and you only measure what you can log. So questions like we have about the offline experience will hopefully get easier to answer as data becomes more ubiquitous.

Good data science is more about the questions you pose of the data rather than data munging and analysis.

Every now and then I see an article about the field of data science disappearing to automation. In effect, the tools get so good that you don't even need to analyze data; the insights are just there waiting for you. While this may be partially true with the growth of machine learning, I don't think it will ever fully be the case. Good data science is more about the questions you pose of the data rather than data munging and analysis.

But with that in mind, I can imagine the field of data science opening up to people that are less technical. As tools get more sophisticated and easy to use, we'll see more people getting excited to work with data. We've already observed this at Airbnb, where we train everyone in the company to use SQL. As I mentioned earlier, you don't want your data science team to be the gatekeepers of all information. We want everyone to be able to interact. I love watching people with no background in statistics or CS wrapping their minds around the basics of working with data. They get so excited, then they get curious. And that frees us up to focus on interesting problems that will impact the business.

This sounds like the democratization of data science.

Exactly. It's happening today at Airbnb, and I bet we see a lot more of it in the future.

CLARE CORTHELL Data Scientist at Mattermark

Creating Your Own Data Science Curriculum

After graduating from Stanford, Clare Corthell embarked on a self-crafted journey to acquire the knowledge and skills to understand and analyze macro-behavioral trends. One thing led to another, and her collection of resources turned into the Open Source Data Science Masters - a curriculum of online courses, books and other resources that one could use to learn the mathematical and programming foundation crucial to a data scientist.

Clare took a risky move by crafting her own degree program, outside of traditional educational institutions. She faced skepticism of a self-taught individual in a job that is typically inhabited by PhDs, but also found a community of supportive colleagues.

Overcoming these challenges, Clare completed her Open Source Data Science Masters and found herself as a data scientist at Mattermark, a venture-backed data startup working with large datasets to help professional investors quantify and discover signals of potentially high-growth companies.

What was your background, before you began the Open Source Data Science Masters and before your role at Mattermark?

I'm a product person and an entrepreneur. I fell in love with startups long before I attended Stanford, where I designed a degree in a then-obscure program called *Science, Technology & Society*. You get to marry two engineering tracks, so I ended up designing a degree in product design and digital development, which then got me started working on product with early stage companies.

Before the OSDSM (Open Source Data Science Masters), I was designing and prototyping products for an early stage education technology company in Germany. Designing from user anecdotes alone became difficult when you only pull from anecdotes, so I started digging deeper into analytics and customer profiling. I started thinking about observing meta-trends among users instead of studying their behavior with a clipboard from behind a one-way window. What if I just ran several tests on two different prototypes? Then we would have data to tell us which one to develop! But as with many European startups, the company didn't get funded, so I had a few weeks to think about how this new perspective fit in. On a long layover in Barcelona, I ordered an espresso and wrote

down the technical skills I would need to dissect meta-trends and understand user data. That list laid out 6 months of full-time work, after which I'd really be able to do some damage. This became the Open Source Data Science Masters.

> *I started thinking about observing meta-trends among users instead of studying their behavior with a clipboard from behind a one-way window. What if I just ran several tests on two different prototypes? Then we would have data to tell us which one to develop!*

As with any story, it is now retrospectively clear that I would secretly fall in love with an applied statistics class I cheekily called "Exceltastic." We worked with Bayes' Theorem and Markov Chains in the business context, figuring out things like how many cars can pass through two toll booths per hour. Everyone else sulked and moaned through munging spreadsheets while I harbored a dirty secret: I loved Excel models! Even so, I didn't know when my toll booth throughput calculations would be demanded of me, nor what class logically comes next. It took getting into industry to shed light on the value of keeping metrics. Things like my Exceltastic class don't seem to fit into an overarching puzzle, but we believe they shape our path. That's the power of confirmation bias. One of my favorite designers has this phrase that he prints in various media: "Everything I do always comes back to me." I've always found that fitting.

What is the Open Source Data Science Masters? What does its curriculum look like?

It's a collection of open-source resources that help a programmer acquire the skills necessary to be a competent entry-level data scientist. The first version included introductory linear algebra, statistics, databases, algorithms, graph analysis, data mining, natural language processing, and machine learning. I wrote the curriculum for myself, then I realized that people all over the internet were asking for it, so I published it on GitHub.

In August, I opened the curriculum for pull requests on GitHub. Without feedback it's difficult to know whether you've covered the right things. Further, it was an effort to get feedback on the idea of an institution-free degree, a kind of home-school for advanced degrees. The internet was astonishingly supportive and excited — and that excitement is addictive. It makes you want to be more transparent, and to become part of other peoples' wonder in learning new things.

How did you get started with the Open Source Data Science Masters?

I knew that a traditional Masters program would take at least the next three years of

my life, but even more importantly it wouldn't focus on what is core to the profession I wanted to enter. I knew what I wanted and I was willing to take the risk of a non-institution education.

It took getting into industry to shed light on the value of keeping metrics. Things like my Exceltastic class don't seem to fit into an overarching puzzle, but we believe they shape our path.

I set out for the curriculum to take 6 months to complete (March - August 2013), with a small project at the end and various programming mini-projects focusing on scraping, modeling, and analysis. It was amazing how difficult it was to manage myself. School gives you this structure that you don't have to question or design, which you don't really see until you have to manage your own curriculum and deadlines. There's a lot of product management that goes into an educational track like the OSDSM. I'm grateful to all the people who supported me and helped me throughout, even if they didn't quite understand the strange and uncharted waters I was braving to get there.

How did you find the resources?

I reverse-engineered most of it from job descriptions that interested me. This meant companies I believed would grow quickly and provide the most opportunity: mid-stage startups, 100-200 people, existing data science teams and reverence for the methodology. I didn't want to be the lone wolf and knew I needed mentorship.

People tend to frown on centering the goals of the classroom on applicability in the real world, but a classic liberal educational approach in a technical career pivot won't serve you. This is a technical vocational degree, so the goal was very concrete. I should be employable and employed on a data science (or Analytics Engineering) team after completing the curriculum.

There was another realization that coalesced very quickly: the act of designing from insights of single users does not scale. I was also hankering for something more technically and algorithmically challenging. I'd bought this book before I moved to Germany, *Programming Collective Intelligence*. I just bought it, I really had no reason to. When I first opened it up, I understood next to nothing. But I carried it with me in Germany, and every time I opened it, something new jumped out and I understood more about scaling user insight. The book became my cornerstone, how I measured my progress. It's a bible for Data Scientists.

I also used the following resources/websites:

- **Quora**: This is a great resource for the Valley — it's truly navel-gazing, but if that's what you're doing, it's useful. People like DJ have answered questions about what a Data Scientist does on a daily basis. You can start to discern the technical capacities that are required of you, mathematical foundations that are necessary, and so forth.
- **Blogs:** Zipfian Academy, a data science bootcamp, had a blog. They had a great post on the resources they saw as core to becoming a data scientist: A Practical Intro to Data Science
- **Coursera:** I'm Coursera's biggest fanboy. They're part of this quietly-brewing educational revolution, which will soon be less quiet. My story is a tremor before the earthquake, I'm just waiting for the ground to start shaking.

How much math (probability, statistics, ML) did you try to learn? How much math do you think a data scientist needs to know?

You don't have to know everything. That's why I've tried to keep the curriculum so tightly focused on its goal. Programmers are great at "just-in-time" learning because it's impossible to know everything. That's a great trait. If you have a core set of competencies and understand how to "debug" problems and learn what you need to solve them, you can do damage. And naturally, you improve over time by recognizing new problems as chunks of old problems you've already seen and solved.

The internet was astonishingly supportive and excited — and that excitement is addictive. It makes you want to be more transparent, and to become part of other peoples' wonder in learning new things.

So much of this curriculum is abstract, and that's where people get scared. People are scared of math because it's not applied in our education system. But those scary elements of math and abstraction diminish with concrete examples and conversations with others. I had a few phone-a-friend lifelines, and I ate up Khan Academy and Coursera videos. There's something magical about how much more communicative spoken English can be, especially when you can rewind and digest a concept for the second, third, or even fourth time. You can always talk through a problem with someone else, even if they're not an expert. Talking through things is synonymous with debugging. One of my mentors calls this "the rubber ducky method," because if you talk a problem through to a plastic duck, sometimes you start to find the holes in your assumptions. Then you can plug them up.

If you think about people as having different levels of competency in these different realms, it doesn't take long to understand that working as a team allows you to stack

your respective skills on top of one another. Having specialties among the team is really essential to getting things done in a small organization. I was lucky enough to join a company where I get mentorship in verticals where I'm middling or even an amateur. It's amazing to learn with other people. Finding a job where you have mentors and training is essential to continue to grow and improve. And if you're not improving and growing, you're dead in the water. So that's a long-winded way of saying: Working with other people is essential to working with more complex concepts and systems. Rome wasn't built by some guy, and probably not at a weekend hackathon.

What would you do differently if you could redo the Masters?

You don't have to know everything. That's why I've tried to keep the curriculum so tightly focused on its goal. Programmers are great at "just-in-time" learning because it's impossible to know everything.

As Patient Zero of a new type of internet-based institution-free education, I didn't know what to expect. It was impossible to know how I would be judged and whether I would benefit from my experiment. This type of ambiguity usually makes people extremely uncomfortable. It's like leaving a six-year-old in the library by herself instead of putting her in class with a teacher. What is she going to do? Pull a bunch of books onto the floor and see how high she can stack them? Watch birds at the window and think about how wings work? Or is she going to find something interesting and gather books that will help her form her own ideas about the world?

I knew that it would be a risk, but I took a leap of faith and left myself alone in the library. In the end, the greatest reward didn't come from the curriculum, it came from what taking a risk demonstrated about me. It led me to a tribe that respected the risk I had taken, and valued the grit that it required to follow through. Many people were displeased that I let myself into the library without an adult. But I'm not interested in taking the recommended path and clinging to a librarian. I have no interest in small ambition.

What's the difference between data science job descriptions & day-to-day role at Mattermark?

Our CEO Danielle was once asked how many data scientists we have at Mattermark. We're all data scientists, she thought — we all use, manipulate, and analyze data on a daily basis to make our customers happier and more profitable. We even all write SQL! That's not something you see every day at a company, but it's essential when you're building and selling a data product. I build products as an engineer, anything from fitting

clustering algorithms, building automated analyses, designing UIs, acquiring new data — it's a startup. It's all hands on deck.

It's not clear that data science is a job title to stay yet. For example, do we know if growth hacking is a subset of data science? We don't. There will always be a top-level salary for a person who can turn chaos into insights. That won't change. Data Scientist is a title we'll continue to use while we figure it out.

What could someone in school, or otherwise without too much background in industry learn from your experience?

The ability to evolve my own career with a self-designed curriculum begins to outline the immense cracks in the foundation of higher education*. The deconstruction of this system was very long in coming, but it's happening now. The lesson is the following: if you take initiative and acquire skills that increment your value, the market is able and willing to reward you.

The ability to evolve my own career with a self-designed curriculum begins to outline the immense cracks in the foundation of higher education

Though people continue to believe and espouse old patterns of education and success, these patterns do not represent requirements or insurance. The lack of any stamp of approval is a false barrier. There are no rules.

It's important to understand the behavior of the market and institutions with regard to your career. When breaking out of the patterns of success, know that people will judge you differently than others who have followed the rules.

There are two very discrete things that I learned: The market is requiring people to perform tryouts for jobs instead of interviews, and most companies don't hire for your potential future value.

Tryouts as Interviews: The economy has set a very high bar for people coming into a new profession. Job descriptions always describe a requirement for previous experience, which is paradoxical because you need experience to get it. Don't let that scare you, not for a minute. Pull on your bootstraps and get in the door by giving yourself that experience — design and execute on a project that demonstrates your ability to self-lead. Demonstrate that you can take an undefined problem and design a solution. It will give you the confidence, the skills, and the background to merit everything from the first interview to the salary you negotiate.

Even more concretely, work with a non-profit organization (or another organization that doesn't have the economic power to hire programmers or data scientists) to create a project that is meaningful for the organization and also shows off your skills. It's a great way to do demonstrative and meaningful work while also aiding an organization that could use your help, and likely has problems people are paying attention to solving. Win-win.

Current Value vs Potential: Look for companies that will hire you for your potential. It's important to be upfront about your grit, self-sufficiency, and ability to hit the ground running. Luckily, with disciplines like data science, the market is on your side. Sometimes companies can spring for a Junior Data Scientist and invest in your growth, which is really what you wanted from the beginning.

Talk with people who can recognize hustle and grit, and not necessarily those who are looking to match a pattern drawn from your previous experience.

Everyone will tell you this, but I work on product so I'll underline it even more strongly: Learn to write production-level code. The more technical you are, the more valuable you are. Being able to write production code makes you imminently hirable and trainable.

*[*NB: Don't think for a minute that I don't believe in the tenets of a true liberal education - quite the contrary. I continue to read philosophy and history, in part because we cannot draw fully upon the knowledge of man without doing so. These are essential elements to being a purposed, ethical, and effective person - but they don't directly accelerate a career. The true liberal education has nothing to do with market forces, and never should. Higher Education as it exists today and Liberal Education should be held as wholly uniquely-motivated institutions.]*

How was your self-taught path to becoming a data scientist received by company recruiters? What advice would you share with entrepreneurial individuals who are interested in the field?

Talk with people who can recognize hustle and grit, and not necessarily those who are looking to match a pattern drawn from your previous experience. Often, these kinds of people run startups.

Recruiters gave me a very real response: They didn't see my course of self-study as legitimate. It's hard to give yourself a stamp of approval and be taken seriously. I wouldn't recommend that just anyone do what I did — it will take a while for autodidactism to

become more accepted, and maybe it will never be a primary pattern. But maybe people like me can help expose this as a viable way to advance professionally. I know that great companies like Coursera will continue to innovate on these new forms of education, keep quality high, and democratize access.

tl;dr

If you want to get to the next level, wherever your next level may be, it's possible to pave your own road that leads you there. It's a monstrously tough road, but it's your road.

DREW CONWAY Head of Data at Project Florida

Human Problems Won't Be Solved by Root Mean-Squared Error

After graduating with degrees in both computer science and political science, Drew found himself working at the intersection of both fields as an analyst in the U.S. intelligence community, where he tried to mathematically model the networks of terrorist organizations.

After spending a few years in DC, Drew enrolled in a political science PhD at New York University. It was here that he drew up his famous Data Science Venn Diagram. It was also during this time that he co-founded Data Kind, a nonprofit organization which connects data experts with those who need help. After a stint at IA Ventures as their Data Scientist in Residence, Drew joined Project Florida as Head of Data, where he uses data science to give individuals better insights into their health.

Drew is also the co-author of the O'Reilly book, Machine Learning for Hackers.

Your data science Venn diagram has been widely shared and has really helped many people get an initial sense of what data science is. You created it a long time ago, back in 2010. If you had the chance to create it again today, would you change any part of it?

Quite a lot. I can speak a little bit about the history of it which I think is probably less glorious than people know.

I was a graduate student at NYU and was a teaching assistant for an undergraduate class in Comparative Politics. As a teaching assistant in those classes, your mind wanders because you already know the material.

It was 2010, and the idea of data science was much more primordial. People had less of a sense of what data science was. At that time I was thinking about the definition of data science. I had been speaking to people like Mike Dewar, Hilary Mason and some other people in New York and was influenced by their ideas and some of my own and came up with the definition while sitting there in class.

The original Venn diagram I made on data science, which ended up becoming quite well-known, was drawn using GIMP as the editor — the simplest, cheapest program in the world. But I'm very happy that it seems people have attached themselves to it and it make sense to them.

What has become more apparent to me as the years have passed is that the thing missing from it is the ability to convey a finding, or relevant information once an analysis is complete, to a non-technical audience. A large amount of the hard work that most data scientists do is not necessarily all data wrangling and modeling and coding. Instead, once you have a result, it's about figuring out how to explain that result to people who are not necessarily technical or who are either making business decisions or making engineering decisions.

Really, it's all about conveying a finding. You can use words to do that, you can use visualization to do that, or you can develop a presentation to do it. A well-rounded data science team will have someone who is very competent at this. If your organization is making decisions based on your analysis, you need to be sure they understand why.

A large amount of the hard work that most data scientists do is not necessarily all data wrangling and modeling and coding. Instead, once you have a result, it's about figuring out how to explain that result to people.

This echoes parts of what we've heard when we talked with Hilary Mason and Mike Dewar. Both of them emphasized the storytelling part and how to carefully communicate the analysis part.

It's something that receives the least amount of thought, but turns out to be one of the most important things once you're doing this in the wild. Even the people who have had success in data science up to this point have just been naturally good at it, whether they were blogging about it or giving good presentations. Both Mike and Hilary are examples of people who are good at doing that. They are naturally good at it. People who are not naturally good at it can learn about it through coaching, and mentorship.

In just the same way, if you're not a good coder you can become a better coder through coaching and mentorship.

You said on a Strata panel: "Human problems won't be solved by root mean square error." What did you mean by that?

I think when people think about data science, or even machine learning applied to data science, people think that we have a well-defined problem, and we have our data set. We need to find a way of taking that problem and that data set and producing an answer that is better than the one that we currently have.

For example, Kaggle does a really good job of finding a problem definition, finding the data set, saying that it's connected to that problem and ramping up to a competition. That way people can try and achieve a very specific thing such as having a better predictor, or having a better classifier so your errors are small.

But the really hard problems are ones for which we don't have good well-defined definitions for yet. Or we recognize the problem but it's not obvious how to find the relevant data that goes with it. Those are really hard problems to me. I'm a social scientist by training, so I think about how human behavior could be observed, what it is that I want to learn about institutions or policies or government and interventions to help keep a lid on our lives.

Those problems are very hard to model. So they require more creative thinking. Particularly at first, or at the onset where you have no idea if there's even any relevant data out there. You might have to go on and run an experiment, run a data collection experiment. Then try from there. "Ok, what are the models and methods that might work in this context?" At the end, you're going to spend a lot more time thinking, "Alright, what are the intended and unintended consequences that might result by implementing my idea?"

Take New York City, for example. Let's say you wanted to optimize the snow removal routes in New York City when there's a snowstorm. Those who were in New York when there was a big snowstorm might remember — there were a lot of people who complained because the snow ploughs couldn't get to certain neighborhoods fast enough.

So technically it's probably a pretty easy problem to solve. It's like a rough optimization problem. You could do that. But if you take a snow plough that's expected to be in one place and reroute it to another place, the people who live in that block will have a negative effect on optimization. Or at least there will be a perceived negative effect.

This is a long-winded answer, but it's much easier if you're only thinking about minimizing error. If you have a broader perspective on how your application or your problem or the solution to it actually impacts people, it becomes harder and therefore much more interesting and useful to the discipline of data science.

How have you found working at the intersection of social science and data science? What are the problems that you've really chewed on and how did you come to arrive at those sorts of problems?

For me, it started where you are, in my undergraduate times. I was a computer science student but I went to a liberal arts college so I got to take lots of other classes. I always

found questions that were being asked in my political science or sociology classes to be the ones I was really interested in: "How do groups of people make choices? How do markets move? Why is one group of people making different choices than another group of people? What motivates people to do bad things? What motivates people to do good things?" These sorts of questions were much more interesting to me at the time than writing a faster compiler or a different programming language.

If you have a broader perspective on how your application or your problem or the solution to it actually impacts people, it becomes harder and therefore much more interesting and useful to the discipline of data science.

At that stage I actually ended up double majoring in Computer Science and Political Science so I had to write two theses. My political science thesis was back in 2004. Keep in mind that when I went to college, 9/11 was a big part of my experience. So I became really interested in terrorism and terrorist groups. I was reading trying to learn more about it. At the time peer-to-peer file sharing networks were still prominent. I was reading about how those file-sharing networks were used and the way data went through them and I observed that they were structured in very similar ways to nefarious networks or terrorist networks. I wrote my thesis on mirrors between these two things. There are weaknesses in the file-sharing network. If it was possible to replicate those weaknesses in a human network, maybe you could exploit the same weaknesses that people use to try to intercept communications on a file-sharing network.

I actually got invited to present that paper at West Point when I was a senior. This set me on the first part of my career path. I started my career in the intelligence community and there were people at the conference from various intelligence agencies who were really interested in the idea that you could model human behavior in the same way you model computer traffic.

Part of it for me was that I felt a connection to the 9/11 event and I was interested in learning more about why people would do that. So between the knowledge that I had learned in Computer Science and my interest in Social Sciences, I landed a job as a computational social scientist working inside the intelligence community. The problems I was working on there were exactly an extension of the work that got me there: understanding networks, working out how people make choices in non command-and-control structures.

Ever since then, I've always been fascinated by computer science, math, and statistics as a tool belt. I find these technical things really interesting to apply to human problems.

I'm not working in the intelligence community any more, but since then, I've worked on my Ph.D. and have done research in the space, and have even started an organization like Data Kind, which tries to scope out the intersection of where the human problems are, where the technical talent is and then put them together. And now at Project Florida, I've always wanted to do take these learnings and apply them within the sensor market and with healthcare. It's always been the classic problem that's excited and motivated me.

How is it that you were able to come straight out of undergrad and begin working in this domain?

For me, I'm not sure my career path is one that I would recommend for other people. I loved my career, I can't complain about any step. But we'll call it an outlier situation. I was working with a lot of "reformed" academics. The people who were mentors to me had been professors at big research universities and it was very multidisciplinary. I had colleagues and bosses who were PhDs in math, computer science, economics and sociology. I was working with a large group of really smart people.

I started my career as a very junior analyst. The way that DC works, in a sense, is that in order to reach the next "level" you have to have at least a Masters degree. Well, I got to that point around 2007, so I was thinking about what I wanted to do. I was reaching out to my colleagues and mentors for advice. They sat me down and they said I had two choices: "You can do the typical DC thing which is to go to night school, get your Masters degree and then do the next thing. Or you could think about becoming a professional researcher. Go back to school full-time, see if you're interested and do a doctorate."

For me they were saying, "We know you, we know what you like doing. You should really consider the Ph.D. because we think it would be good for you."

To be honest, I didn't really want to do it. It's such a huge opportunity cost. If I did it, that was five years I could have been making money and building a career. However, on their counsel I started looking around at some programs. I knew I didn't want to go back to school for a computer science degree or a math degree, because I definitely wasn't the greatest computer scientist or mathematician that ever lived. And also, at the end those are not the problems I want to solve. If you're going to do a PhD., you have to contribute back to the discipline. I wasn't interested in contributing back to those two disciplines, so I thought about various Political Science programs. I wanted to find one that was very quantitative. I ended up at NYU Political Science, which was one of three or four political science departments in the world that was heavily quantitative right from the start.

It was also in New York.

I felt that being in New York and in a large urban area opened up a lot of different things and wouldn't limit me to focusing specifically on my academic endeavor. I could be exposed to other things while I was there.

I also decided that I wanted to talk more publicly about the work I was doing. Part of this is colored by the fact that for years, by being in the intelligence community, I couldn't talk at all about the work I was doing. So moving on from there, I was really eager to start blogging or going to the media to talk about the work.

As soon as I got to graduate school, I started doing those things. That helped balance the work I was doing as a graduate student with running the Meetup in New York, giving talks, advising start-ups and getting involved. That doubled my work but it was all fun work and I really loved it.

The decision to go back to school was basically, "Well, I think this would be good for my career." I didn't even really know if I wanted to be a professor. It was something I was interested in, but I knew if I was going to become a professor I was going to be the kind of professor that had one-and-a-half foot in the university, and the other half foot out doing stuff.

From my experience at graduate school I decided I definitely didn't want to be a professor. My father was a professor so I'm sort of a university brat. I know the lifestyle is fantastic — there's nothing wrong with it. However, the realm of a university is teaching and publishing and not building software or data science.

Given that you had the experience of working in industry before going back to graduate school, do you feel that you had a significantly different perspective? Were you looking at the academic problems you were facing in grad school differently because you've had a chance to dig your teeth into them already in the "real world"?

One thing I always say, and I tell this to people all the time, is that I highly recommend not going directly from undergraduate to graduate school. Even if it's just to work for a year, I think it provides you so much more insight and experience in the kind of problems that are interesting to industry versus the problems that are interesting to researchers.

My early industry experience was unique in that the work I was doing in the intelligence community was split between two halves. One half was the classic intelligence aspect: studying people for short-term projects that have to be turned around in a very narrow time window.

The other half of my job looked much more academic. These were long-term research projects; we were working with specific agencies that had the capacity to do high-risk research. Through that experience I decided that I really enjoyed and was interested in solving hard problems. One of the problems we worked on is how to enable non command-and-control structures (e.g., organizations without coherent org charts) to make choices.

I highly recommend not going directly from undergraduate to graduate school.

For example, in a command-and-control organization like the army, if you're Lieutenant Colonel and you're promoted to full Colonel, everybody understands how that works. However, when you're in a non command-and-control structure, different people in each part of the network have different responsibilities. One does fundraising, one does surveillance, and one does operations. Suddenly there's a person from the operations cell who gets captured; how does that operations cell make a choice about who will become the new leader? Or does someone get taken from another cell and worked through the system that way?

We've thought a lot about how to solve that problem and we didn't solve it at all. However, I got really excited about the thought of solving longer-term problems. So I had another reason to go to graduate school. There was a lot of freedom to think about solving problems that I found interesting.

I think the basic difference there is in industry is that it's about always solving someone else's problem for them. Now, that's not an absolute truth, but certainly when you're starting your career you're almost always solving someone else's problem. Then when you get to graduate school, you get to think of those problems on your own. The issue is sometimes those problems are really boring or they're not interesting because you don't have enough experience or enough knowledge to recognize good problems. That's where mentorship as a graduate student becomes important.

If you're going to go to graduate school, you've got to trust in and work really hard with your advisor because if you don't, you're probably going to produce bad research. It's way easier to produce bad research than it is to produce good research. In industry the objective function is set by someone else; that objective function typically is profit and the problems are usually smaller and more attainable.

So if you have experience on either side of that, you can be more reflective about how it might be on the other side. I think that there is a certain strictness versus a freedom component and there's positives and negatives on both sides. It's really about what

motivates the person doing the work, what kind of stuff you like to do, and how you see your own self-worth measured in terms of what you're contributing. Because in either case, you're never truly independent. That's a fallacy that's built in at graduate school.

In reality, you're the furthest possible from being autonomous during graduate school. You're certainly more autonomous than you would be working for a big company on a team. But you have many masters as a graduate student, the least of which is yourself, and you have to be really good at maintaining your own schedule and solving a problem on your own.

Your book "Machine Learning for Hackers" is in the canon of data science now. Given that, can we talk about the tools that you have found to be useful in your career and also while doing data science? How do you discover useful tools for data science work?

Personally, I am not as much a lover of languages as some computer scientists are. Have you ever heard of the Strange Loop Conference in St Louis? It's in St Louis every year; it's a fantastic conference and I highly recommend it. But it's for people who love tools and love programming. So I went there and was doing an introduction to machine learning programming. I found I was very much a fish out of water there. I was surrounded by people who I respect and who do interesting work and all they cared about talking about was the hot new programming language.

So my approach to tools is: is the cost-benefit of me taking the time to learn the tool going to have a significant impact on getting my work done more efficiently or effectively?

For example, I'm now known as an R programmer because my book heavily uses R. The truth is, I'd never written a line of R code before I got to graduate school. I was a Java, Python, command line programmer from undergraduate, along with a little bit of MATLAB. When I went to graduate school all the statistics classes were taught in Stata. It's a point-and-click statistics program and you have to play by the rules. Eventually what the program allows you to do is, you have to use this highly stylized, domain-specific language for Stata called Mata. During graduate school, we were writing our own optimization functions in Mata. I was looking at the syntax and I didn't know how I was going to do it. It was so far afield from any relevant training I'd had in computer science. So I raised my hand and asked, "Can we do our problems in R?" And the guy teaching the class said, "Sure, I don't care."

Since I'd never programmed anything in R, I set out to teach myself how to program in R simply so I could finish my problem sets for my Intro to Statistics class. For me, once I'm committed to doing it I really want to learn it all and go really deep.

I want to point out that I'm building up a little bit and giving you a false sense of the binary nature of this tool choice. I'd known for a while that R was a language that had a lot of things in it that would be very useful for me. But it has a very tricky syntax and it's not designed well as a language. So it's a little bit of a steep learning curve at the beginning, but once you get over the hump you can do these wonderful things very quickly.

In New York, the anchor industry has always been finance, media, advertising, entertainment and to a certain extent, higher education. Those anchor industries have always been about data.

It was the same thing for me with JavaScript. No one in their right mind will ever ask me to build a website. I don't do that, it's not what I'm good at. But, I got to learning it eventually when I was blogging more often. I was so tired of posting an image file of a graph I'd drawn; it would be much more interesting to have some interactivity where the image or the graph wouldn't just answer the first order question: what is the structure of the data? But that it could also answer the second order questions: Who is this point? Why do we see what we see?

My entire motivation for learning JavaScript was so that I could use d3. All I cared about was being able to create interactive visualization. There's a useful tool out there that I don't understand how to use, so I'm going to learn it so that I can use it. And now, for me, — the worst JavaScript programmer in the world — everything to me is a d3 problem. You could ask me to create a simple online form where I'm collecting your address and I would use d3. I don't really understand any other way.

I learn something through trying to solve a problem. In that process I brute-force my way into having a better understanding.

I'm the same way with mathematics and statistics: I learned probability theory, calculus and linear algebra. I was interested in solving a problem and those were the tools I needed to learn. I didn't have a pure love for those things. Some people love math and love to learn about math. I think it is beautiful, but I'm not an artist. I'm more of a mechanic.

I think that's powerful and pertinent for people who feel they can't get started doing the things they want because they haven't checked all the technical boxes. It seems like another way to do it is actually go into what is the problem you want to solve. Since we cannot solve the problem because of a particular tool or medium of expression, then go and learn how to do a particular part. You're always told to solve the problem first.

That was our motivation for "Machine Learning for Hackers" too. People who are sitting

in a job who are now being asked to run a classifier are asking themselves: What's a classifier? One way to learn what a classifier is, is to go and read Hastie and Tibshirani, or some classic machine learning text, and try and beat yourself up over all the notation in that book. 90% of people don't have the time or don't want to have to read that. A better way would be to say, "Here's a problem you're trying to solve. Here's how you solve it. Here's a tool that will help. Let's open up that black box a little bit. Explain to you a little bit. Not talk about the math. If you care to learn there are other references, you can go to. But it's not a must. So "Machine Learning For Hackers" is written around 12 problems that we try to solve. That was the motivation. It was like writing a book that I wish I'd had before I went to graduate school.

I will say it's an exciting time. There's a lot of opportunity for people to build that landscape. It's very early days; people still don't know what we're talking about when we say data science exactly so there's a lot of opportunity.

What are the exciting data-related things that you're seeing right now in NYC? How's the data ecosystem evolving in New York, and what are the parts of it that you find exciting?

I'm very biased in thinking that New York is the best place in the world to be doing this work. The reason I think that is, if you look at the history of any big city, they have anchor industries that by and large define the city itself. You can look through a list of American cities and see that. If you look at Silicon Valley — the technology industry has always been the anchor for Silicon Valley. There, the focus has been on innovation, software engineering, hardware engineering and how to build better machines, better pieces of software.

In New York, the anchor industry has always been finance, media, advertising, entertainment and to a certain extent, higher education. Those anchor industries have always been about data. As a result, what started as a nascent community in New York has gotten bigger and bigger and has been heavily influenced by the fact that everything around you that's happening is pivoting off of data; that's how everyone in the city makes their money. So there's billions and billions of dollars that go through New York City every day that are really a function of data science.

Thus, the data science community in New York benefits incredibly from its history. Now we have people who care about writing software, which is different from the same anchor industries that have existed in New York. However, these people still benefit from the huge talent pool and the huge amount of money in the city. Therefore it's no surprise to me that the data science community is growing very, very quickly. People are moving here to do this work because in a sense it's always been here, it's just now that people

are paying attention to it more, because it's not the boring anchor industry that we've always known about.

The other piece that I'll say for New York as opposed to other places is that we benefit tremendously from our geography. For better or worse, Manhattan is a tiny island that seven million of us live on. It was easy for me when I was in NYU to take a subway up to Columbia or walk up to Union Square. It really galvanized our community because people were just close to each other. I could have lunch with Mike Dewar if I wanted. That's great.

Whereas if you go to other places, particularly Silicon Valley, it's just so geographically spread out that if I worked in San Francisco and I wanted to go out for lunch with someone from Mountain View, it's an hour-long drive.

Likewise, if I wanted to go to a meetup in San Jose but I worked in the Mission district, it's a pain in the ass. You can't do that. So it becomes much more disparate out there. If you look at the community as it exists out there, it is very broken up. I think that hurts them because community for data science is really all about sharing ideas.

Sometimes others say to me, "You're so unique, no one else can make the transition from social science to data science today?" That's absolutely wrong.

It's much more collaborative in that way. I think New York has had a history of that through different industries.

Right. The density of the networks that you are interacting with is a huge factor in terms of the information exchange of ideas and how cross-disciplinary you can be.

We tried to institutionalize that with Data Gotham in a sense. Data Gotham is the conference that Hillary and I were doing, and people seemed to like that. Now there are other geographies that are trying to do a similar thing. For example, DC has got one. Similarly, there are big data science conferences in Silicon Valley.

You gave a talk recently where you made people stand up and promise to hire more social scientists. What advice do you have for people who have both a social science and a computer science background and who want to go into data science?

The piece of advice that I would have would be to continue following this track. You're a social scientist and you care about human problems and the specific genre of those problems that triggers your interest. If you have a desire to solve a problem from the

world of social science using the skills of your computer science, you need to dive pretty deep into whatever the technical tool is that you care about. I talk to a lot of social scientists who are thinking about learning Python or R and they're not sure which one to pick up, but just dive deeply into one of them.

It doesn't make any difference. Just pick one, use it and learn from your mistakes, but make sure you're asking intelligent questions.

You're either trying to learn something new or you have an interview or a question that you ask that you don't know the answer to and you can say, "I tried this, but I wasn't quite sure so I went back and tried something different."

A piece of motivation I would give people is that sometimes others say to me, "You're so unique, no one else can make the transition from social science to data science today?"

That's absolutely wrong.

The problems that you care about, people will pay you lots of money to work on. Every way that an internet company makes money is by humans making choices; the choice to buy something, the choice to click on something, to share something, to connect with someone.

All those things are questions that are fundamental to the social sciences. So you already have all of the training necessary to identify the problems that are out there in the real world. Now all you have to do is figure out how to solve them using the tools from an industry.

Don't think you can't do it because, the reality is that you're already way ahead of the game. Now you have to learn the easy stuff. The hard stuff you already know. Go learn these things, and then get better at it.

KEVIN NOVAK Head of Data Science at Uber

Data Science: Software Carpentry, Engineering and Product

Kevin trained as a theoretical nuclear physicist where he used statistical methods to evaluate theoretical models for nuclear interactions. It was during graduate school that Kevin realised that he liked solving difficult problems, but not in an academic environment. A friend from undergrad came calling and soon Kevin found himself applying his skill-set to solving the mathematics of logistics.

Today, Kevin is the head of data science at Uber, where he leads a team collecting and analysing a vast array of data within the Uber global network to inform product decisions and better serve clients. He talks about the importance of a relentless curiosity to solve problems, and the need to develop a well-rounded suite of skills across engineering, data, and product.

Let's start off by talking a little bit about your background.

I am a Senior Data Scientist at Uber and run the dynamic pricing group today. I have worn a lot of hats during my time at Uber and have been with the company for about 2.5 years. I'm the second full-time data person and 20th employee at Uber.

What were you doing before Uber?

Before Uber, I was a Ph.D. candidate in nuclear physics at Michigan State University. I was there working on the cyclotron in the theoretical physics department. Anything theoretical, but especially physics, requires a lot of computer programming.

It's a whole, long involved process, but essentially involves using statistical methods to evaluate theoretical models for nuclear interactions. We then evaluated the models based on the output data from the particle accelerator.

We evaluated if models can be confirmed by experimental data.

What got you interested in data science?

I always have been the bad physicist. I have always used computing tools versus using an experimental setup. In undergraduate studies, I wrote a program to build computer-

generated holograms. As a physicist, I was always a bit different and did a mix of theory and computer science.

When I went to graduate school, I quickly realized that academia wasn't the best fit for me. Academia wasn't what I wanted to do for a career. I wanted to do something different, but my background was weird during those times when I had a highly specialized focus and a wide suite of computer programming skills. It was hard for the middle 80% of companies to find a good fit for the skillset that I brought.

It was hard for most mainstream companies to justify hiring a nuclear physicist for a job that is not nuclear physics. That's true in most specialties.

I got a call from a roommate in undergrad, who was an early engineer at Uber. The job description required the ability to write production code and to be good at mathematics. It was the perfect fit for someone with my background and so I decided to join immediately in June 2011.

You mentioned earlier that the role at Uber required a mix of computer science and mathematics, and that other companies simply didn't know where to put you. Could you elaborate on this?

It was hard for most mainstream companies to justify hiring a nuclear physicist for a job that is not nuclear physics. That's true in most specialties. I didn't even know it was called data science when I started out, and realized that data science was a buzzword that was rapidly growing up.

Data science encapsulates a skill set and a style of background. Almost everyone at the Uber data team is from a nontraditional background. Everyone here was doing something different at some point in their lives.

This unconventional transition for most data scientists may change in the future, but just having the hackerish mentality and flexibility is very relevant for aspiring data scientists. This ability to cross-pollinate ideas is especially relevant for startups where you are expected to wear different hats.

So you mentioned a little about what you think data science really is. If you had to boil down the role and purpose of a data scientist, what would that be?

Data science is rapidly becoming a buzzword with all the positives and negatives associated with that. It helps to encapsulate a series of broad ideas as a rallying point for individuals like myself.

At the same time it can be easy to hype a concept without actually having a strong understanding of what it is. In my opinion, the field of data science really has two main specialties. One is the concept of "big data", where large amounts of information are processed to derive mathematical insights. For example, Twitter and Facebook are famous for the products they've developed using this work style.

The opposite specialization in data science (probably closer to my job) is a more highly specialized predictive modeler, where there is a need to make quantifiable decisions based on heterogeneous pieces of information. For instance, predictions based on incomplete information from a sales representative and information from another company which had done this before. These sort of predictions require a considerable amount of programming, statistics, and mathematical intuition.

This unconventional transition for most data scientists may change in the future, but just having the hackerish mentality and flexibility is very relevant for aspiring data scientists.

How much of your time is spent cleaning data vs. doing actual analysis?

Cleaning data is very different for the two branches of data science that I just mentioned. On the larger end, some statistical errors are negated by the virtue of having a lot of information — the Law of Large Numbers. Everything converges on a normalized distribution; statistical anomalies will very rapidly disappear.

On the other end, when you are trying to do predictive modeling based on a small set of incomplete information, one outlier can quickly throw your prediction off if you do not have a solid understanding of the process or problem that generated it.

The cleaning process is very different between these two regimes. On the smaller end, it is more a matter of evaluating the confidence one has in one's data, while on the bigger scale, it is more about building up a more homogeneous data set to feed into algorithms.

One of the most rapidly changing areas of the field is cleaning data. There are more data science and numerical computation toolkits out there than there were 18 months ago. Where these toolkits shine is in their simplicity in allowing large amounts of information to be thrown at them.

The operations are fairly simple in terms of cleaning data, but what's challenging is scaling these solutions to very large data sets.

What are some of the most valuable tools and the most valuable skills that someone should have if he/she wants to work in data science?

A lot of people have this biased emphasis on algorithms and programming languages. A lot of the programming languages and problems that we worked on 20 years ago are very similar to the problems that we face today. In the big data regime, algorithms that can scale to massive data sets to deliver quick feedback already exist in closed form. So the algorithmic solutions already exist. Where you get paid as a data scientist is the skill in constructing a data pipeline to feed into algorithms and knowing how to apply those algorithms in specific contexts. These skills are all derived from mathematical and statistical intuition.

So a rudimentary understanding of mathematics and statistics will get you 85% of the way there, while the last 15% will come from basic coding skills. A statistical background and intuition will get you a long way.

So a rudimentary understanding of mathematics and statistics will get you 85% of the way there, while the last 15% will come from basic coding skills. A statistical background and intuition will get you a long way. We're not in academia anymore and can just skip quickly to the solution.

You just mentioned this 85/15 split and for a lot of people that we've spoken with that have the adequate background, there's this fear that they are not adequately prepared in terms of programming and work experience. A lot of people are concerned that they don't have the relevant practical skills to transition into data science. Could you speak a little to their experience?

Different companies have a different opinion about what engineering and statistics background is required for data scientists. At Uber, the data team is fairly engineering-oriented and we do a lot more implementation than a typical data science team. At a lot of companies, data scientists are part of the business or product team and as a result their work is a lot more qualitative, which obviously informs the job requirements.

We have to be able to write computer code in order to solve mathematical problems on a computer. Having the ability to write professionally organized software code is a secondary skill for data scientists at Uber.

Whenever I talk to other data teams, I always ask where the data team is on the organization chart, and that will tell you a huge amount about the implicit skillsets they expect you to have. Software engineering is a non-trivial part of our job as a data scientist.

Both a statistical and programming background are valuable in different ways. You are hired for the programming, but the statistical background is relevant for elevating you to the next level. So it's really a tradeoff between these two skills which are both valuable.

You recommended for people to take a look at where data science sits in the company structure. So here at Uber, what is it that you do that creates value to the company?

We are at our core an engineering team. In most startups, that setup is fairly common. A lot of what we do technologically is backstopped by data. At the end of the day, Uber is a company about logistics, about getting stuff to people quickly; all of that is a math problem.

Whenever I talk to other data teams, I always ask where the data team is on the organization chart, and that will tell you a huge amount about the implicit skillsets they expect you to have. Software engineering is a non-trivial part of our job as a data scientist.

On the flip side, the data scientists at Facebook or LinkedIn are a part of their product teams. At the end of the day Facebook is about connecting people and while the data component is a nice add-on, it is not a core functionality of the company. Data informs how they scale the company, but it's not an engineering problem. So the requirements for a data scientist are different between a company like Uber and a company like Facebook.

So how do you define personal success?

I'm a data scientist and I'm also an engineer. At the end of the day I want to solve problems. So if I can solve problems today better than I could yesterday, then that's a success.

For somebody who gets into data science and realizes that it's not for them, what can these people transition into?

A solid understanding of what's not working will inform the direction of transition better. Data science is at the confluence of computer programming, mathematics and communication as part of the work structure.

If you don't like mathematics, a better and more obvious role is to get into business development of product.

On the other hand, if you like the mathematics but don't like programming, an analyst position may be more suitable. Some people are arguing that a data scientist is an evolution

of the analyst, but I believe these two roles are on fundamentally divergent paths. An analyst is someone who is answering more financial or quantitative information using an existing toolkit. A data scientist is more of a mix of software carpentry, engineering and product.

If you are good at mathematics and engineering, but not good at communication, I would recommend becoming an engineer. There are a lot of organizational charts in a lot of companies where the engineers are isolated from the other departments. A lot of companies can offer that sort of environment where an engineer can just focus on the problem at hand.

So you've been at Uber for more than two years now and you mentioned this divergent path between the data science and the analyst roles. It sounds like you've had a lot of time to see how it evolves. Broadly speaking, what are the qualities that separate the amazing data scientists from the rest?

I'm amazed by people who are intuitive about problems they have just heard about. For example, Josh Wills is a guy who has never seen my data set, and has only ever heard of my problems through media sources. Josh is someone who can come in and, off the top of his head, reverse engineer the statistics of how people are behaving.

Having that sort of intuitive problem skill is very important and on top of the approximation skills will get you 90% of the way there to being a top data scientist. The other interesting skill set is the ability to work and execute on largely open green-field projects which would take an average team much longer to do.

Having that sort of intuitive problem skill is very important and on top of the approximation skills will get you 90% of the way there to being a top data scientist.

Again, a solid understanding of what's important and how to build your toolkit is the base for making you a rock-star data scientist.

The open green-field approach sounds a lot like the way in which academic researchers approach open-ended problems. What is the difference here in data science versus academia?

The perceived limitation of academia is that they don't have the flexibility to go ahead and do it. At the end of the day, academia is about understanding problems whereas data science is about solving problems and moving on.

What attracts me to data science is the ability to step on the gas and just go. I can solve a problem and move on as long as the relevant people involved in the decision-making understand the solution. The approach is more like ready, fire, aim versus a more methodical process in academia.

In academia, one can work on open-ended problems indefinitely with no expectation of results. How have you made the transition from the academia mindset to a more results-driven environment?

What attracts me to data science is the ability to step on the gas and just go. I can solve a problem and move on as long as the relevant people involved in the decision-making understand the solution. The approach is more like ready, fire, aim versus a more methodical process in academia.

There are deadlines and very non-trivial ones. I have personally been blessed by having one of the most leading CEOs in the industry. Travis is a data-nut, he loves talking about all sorts of problems. Early on he instilled a very experimental culture towards data science implementations.

One of the examples of this was an experiment we wanted to run where I wanted to build a test bed to test out different hypotheses. Travis told me to put it into production to test it. That to me exemplifies the entrepreneurial attitude compared to academia and speaks to the whole *ready, fire, aim* concept in entrepreneurial environments.

In academia, the approach would have been to spend a large amount of time doing meticulous contingency testing in order to come up with the best solution.

More forward looking, for you personally, what would you say are your personal goals as a data scientist over the course of the next year?

I think we are at a pretty cool time in data science, where data science is on everyone's radar and we are over the initial wave of hype associated with the industry. We're still at this phase where 80% of the promise of data science is still unfulfilled.

The leading companies in data, at least in terms of public perception, are still involved in the social space. In my case it is the problem of "*how do I give you a car faster?*" In the grand scheme of things, the problem space that is being tackled with data science is still very open and expanding.

At Uber, we are solving the mathematics of logistics, but one can easily port the same solutions to solving the logistics problems of the world. For instance, what if one could

use data science to give you an ambulance to your location much faster than before? So when one takes a step back there are immense opportunities in the direction we're moving in for data problems.

So the promise of data science is still very much the tip of the iceberg and that to me is very exciting. To me the first part of unleashing this promise is to start building a community of data scientists to enable sharing of ideas.

You mentioned that we are on the tip of the iceberg in terms of applying data science to solving problems. What are some developments in the field that you think are emblematic of this trend?

Every person in data has their own pet project — something they love which they always talk about. I was talking to someone about genomics. There's a really exciting development with algorithms where we can analyze genomes literally as quick as they come out of the machine. The speedup in analyzing genomes is hugely exciting in terms of the possibility of understanding our world.

Nothing convinces like success. If you can find a problem and solve it, or even implement your own solution to common problems, that is how you get people excited.

Problems in the healthcare space represent a huge data promise. It would be amazing if a doctor could just diagnose a patient without waiting for a lab test that takes two weeks.

Another exciting area is logistics, we touched upon it with the ambulance example, but what if one could get an instant delivery without having to wait 3-5 weeks for it to get shipped?

Are there any other final pieces of advice which you would share with people looking to transition from academia to the data science industry?

Nothing convinces like success. If you can find a problem and solve it, or even implement your own solution to common problems, that is how you get people excited. Trying to pigeonhole it to specific problems is not the point.

Just solve problems. Start applying data to real life and the rest will follow.

CHRIS MOODY Data Scientist at Square

Astrophysics to Data Science

Chris Moody started off his journey towards data science by peering off into distant galaxies, studying computational astrophysics at UC Santa Cruz as a graduate student.

As the data revolution hit the fields of science, however, Chris found himself having to learn how to use more sophisticated tools that could process more data. He dove into programming and contributing towards open-source astrophysics projects.

All this culminated in a data science fellowship at Insight Data Science. After completing his Fellowship, Chris joined Square's Data Science team. After leaving Square, Chris is now a data scientist at Stitch Fix, a fashion startup.

Thank you very much for being with us, Chris. Can you tell us a little bit about your background?

I went to Caltech as an undergrad to study Physics. There, I had projects that were largely computational.

For example, a project I was involved in was looking at dark matter simulations. Basically, we don't know that much about dark matter, but we can guess at things that it could possibly do. One of those things is that it could decay. If it decays, the dark matter particle gets a kick, and it goes off in a random direction at a random speed. Galaxies are sitting at the bottom of a gravity well; they're like bread crumbs in a big bowl of dark matter. If the dark matter were spontaneously decaying and getting lots of extra energy, it could popcorn out, and totally change the profiles of galaxies in an essential way. This was a strongly computational project that taught me many skills.

After Caltech, I came to Santa Cruz for graduate studies, still working in computational astrophysics. While I was there, I was doing all sorts of things pertaining to galaxies. We would look through the Hubble Space Telescope at the youngest galaxies in the universe and notice that they were not at all like the galaxies today. Galaxies today are beautiful spiral structures. But when you look back at the youngest galaxies, they are lumpy and clumpy… they look like soup.

So, one of the questions was: Does that mesh well with our ideas of how our universe

formed? We started to look at the simulations and realized that what we observed through the telescope is what we were seeing in our simulations. We were super surprised at these theoretical predictions coming true!

I think the romantic, public idea of a scientist is that you jump into a cave and then five months later, you have a "Eureka" moment and you come out. Then it's glorious. But that's not really how it works.

The next part followed the standard trajectory of a lot of businesses. We got one or two really positive examples of galaxies matching our predictions, and were very excited about the progress. But it was only one or two examples; we wanted to know if this was statistically significant, and so we started to scale up our data. We exploded from 100 gigabytes to hundreds of terabytes of data. This all started at the NASA Ames supercomputer.

It turns out that it's really hard to answer simple questions when those questions don't fit onto one computer. So we had to scale up a lot of our algorithms, and build our own infrastructure and framework. It was at that point that we started to get really interesting results. We started to see that this is generally true, and this attracted a lot of people to our project, scaling up our people power. So we'd get other new graduate student astronomers and explain, 'This is how we work; this is how you can be efficient.'

I think the romantic, public idea of a scientist is that you jump into a cave and then five months later, you have a "Eureka" moment and you come out. Then it's glorious. But that's not really how it works. The reality is: you have lots of bugs, you make lots of errors, and you have to work as a team, which means you have to be able to work efficiently. You have to know how a pull request works. You have to know how commits work. You have to know how to document. You have to file bugs and report to issue trackers. You have to do all of these things.

At the end of all that, I realized that I most liked working with data. I liked working with algorithms. Actually, I absolutely loved working with algorithms.

I spent more time reading about how the algorithms worked and how they found all this truth, despite all the noise and red herrings in the data. I loved doing that and working with people on a project together. It was great. I thought galaxies were cool, don't get me wrong, but I liked algorithms more.

It sounds like you spotted a project, saw that it was interesting and used your experience of working on it to explore your interests. How did your background in

science inform your work as a data scientist?

Science is getting harder to do. It's harder to do it individually and it has to happen as part of a team; a collaborative effort, so we can measure different things. Looking at papers from 50 years ago: having a paper with 50 authors on it was ridiculous, that just never happened. Half the papers out there were published with only one or two authors on them.

Now, that's ridiculously absurd. I can't remember the last time I read a paper with only one author on it.

This means that the data and ideas are starting to grow much larger than one person can manage. In turn, it means that you have to learn how to work with other people. So that's a paradigm shift of science.

It's just because the instruments you have to use are larger. We end up having to use supercomputer resources or we have to use the Hubble Space Telescope to get somewhere. This means that the data and ideas are starting to grow much larger than one person can manage. In turn, it means that you have to learn how to work with other people. So that's a paradigm shift of science, and also something that I think industry has been familiar with for a much longer period of time.

At the same time, a lot of my exposure to things like software engineering best practices, or even computer science, was completely self-taught. I didn't take any formal classes in these fields.

That's really interesting that it worked out so well, and also that that didn't hinder you.

I think that's actually pretty normal. Look at some start-ups. They're really interested in finding someone who can actually do the work; someone who is trying to find and build a whole community and foster that growth. Take that person from the 90th percentile and just teach them the remaining 10% of the small skills needed. These startups are basically instilling habits; thinking about what you're going to do and how it's going to reflect on everyone else in the network, instead of being an isolated person.

Sometimes, that has to happen as a feedback reflex. You have to think of how you're going to fit in with everything else. You have to think about how your code is going to be used by others. I was lucky in that I had a community leader in my project who was really interested in teaching everyone else how to work together, and I learned a lot from him.

Of your friends and peers from Caltech, many of whom have also gone on to do heavy computational physics research, have you found that a substantial portion of them are heading towards industry?

Yes, especially in astrophysics. I can't tell you how many plots I have seen in the last year with the number of faculty jobs remaining constant, or maybe even slightly decreasing with time, compared with the sky-high number of post doctorates. That means that the likelihood of a post doctorate job opening is going down at a ridiculous rate. Even when I was in graduate school, the expected number of postdoctoral candidates went from two to three. If it kept going at that rate, by the time I'd finished my first post doctorate, the expected rate would be four postdocs to every one position.

Clearly, there's a huge supply of post doctorates and not that many positions within academia.

How much did those academic job statistics influence your decision on what you wanted to do after graduate school? Did you feel you could get the same intellectual stimulation from problems in industry as you received in academia?

Yes, it was a hard decision, but you look at it and think, 'How many times do I really want to roll the dice? How much do I really like this?' That fear of not finding a job really destroys a lot of the romance of science. I feel like a lot of people start doing science because they have this romantic notion of becoming the best scientist, or contributing in a noble way. But the truth is that science is a shitty ride.

You can do a lot of the same things that science will let you do, but you don't have to do these things in the world of academia. You can work on science in industry. When I made that realization, and understood I could do a lot of the science, and be involved in a lot of the cool stuff I'd tried to do in the first place, it made me realize that I could switch to a new job outside of academia. At the same time, I didn't feel that I was giving up on what drove me initially. There are a lot of startups that are changing the world, so instead of trying to define clumps and galaxies, I could try to actually work with somebody, and try to change the world. I thought this was really cool and super exciting.

So then you joined Insight — a six week long Fellowship for PhDs looking to enter the field of data science. How much of what the Fellowship taught you would you say was new to you?

All of it. There's a paradigm shift from science and industry. Everything in science is about a fully detailed presentation of an idea; exhaustively explicating all of the caveats. All of the communication is bordered on fully defined facts, or at least as much as possible.

You look at the borders of your project, the borders of the results, and you know the downsides and you know the upsides, and that's because you're terrified that someone will find a deficit in your project, and then nail you for it.

But then the opposite is true in business. The biggest problem is that people have very limited bandwidth. It takes a lot of effort, and there are a lot of people demanding it. So the crux of everything in business is actually being able to move all of your results in as terse and precise a fashion as possible.

You don't need to delineate all of the possibilities, you just need to say what is the major point, and you can go on from there. So, a lot of what Insight taught me was that you need to condense all of your results down as quickly as possible. You get someone's attention and you go; that's the hardest part. As scientists, we were taught to give an hour-long lecture on our project. We didn't have to consider whether our audience was being entertained or not. If they're not interested, you don't care. They're not your audience if they weren't interested in the first place.

Everything in science is about a fully detailed presentation of an idea. But then the opposite is true in business.

It's the opposite idea during the Insight Data Science Fellowship. You have to go out and you have to make every single connection for yourself. You have to boil it down and make it completely convincing that everything you're saying is relevant to them, and you have to do it in 5 seconds. Everything is an elevator pitch. Every YC Company has to give demos in 180 seconds. So Insight is all about building a demo in those 6 weeks, and then pitching it in 180 seconds. You're basically pitching yourself as a candidate to those companies. You're saying, 'Don't look at me like a graduate student. I'm actually super goal-orientated, or systems-orientated. I can take all this data, apply these algorithms, and give you some amazing results.' That's what those three minutes are for, and that's the whole paradigm shift. Now, the focus is not so much on the new idea or how much you've added to the body of knowledge. The focus is what can you tell me in 100 seconds. That's all the CEO has time for.

In scientific lectures, you're not trying to reach a super-broad audience. In the case of science, you're trying to deliver an idea, and then you try to back it up in 15,000 words. You need to do that in business as well. You need to be able to take your idea and defend it. The thing is that, here, you're no longer trying to defend it to the CEO, you're no longer trying to defend it to anyone else. You just need to defend it to yourself, and then you need to give them the ideas; there's an implicit trust there.

No one else is going to check your work and no one else should check your work. You

need to be an independent party and you need to break it down as to what is important.

You have to build up small kernels of truth, and that's all you can deliver. A lot of the time, people find it distressing, but I thought it was great. I thought it was an awesome challenge to be able to compress my message down and figure out what all the tidbits are. It's like a whole design philosophy. I liked the idea of throwing out everything except for what you need to function. I like it from a designer standpoint and also from an algorithms and data analysis standpoint. I think that embodying that philosophy was the single most successful part of the Insight Fellowship.

"Data science" has now become a very common phrase in many business sectors. Yet, it's still nebulous and no one is really sure what it means. So, what does data science mean to you? How would you break it down?

It means a lot. It always means to measure data, being able to make sense of that data, create models of that data, and most importantly, to be able to communicate what that data means.

I think data science splits into two fields, and I believe a lot of hiring companies are starting to reflect this. Data science is starting to break off into descriptive analytics and predictive analytics.

Descriptive analytics is, 'we saw this trend.' Or, for example, 'We saw this spike or dip… is that because our service crashed? We saw this huge spike…is it a multiplicity of things?' It's always asking questions of dynamics, and then asking what is going on. So the raw data comes back, and then you make something useful — actionable business intelligence – from that data. That's descriptive analytics, taking data that has been produced and trying to make head or tails out of it, to drive some decisions out of it. So that might mean, 'We saw some really exciting events in Bulgaria, but why is our site exploding in Bulgaria and nowhere else?' You may find out that it's not really from Bulgaria, or that it's raining everywhere else, or a volcano just went off and everybody's Tweeting about it, or something ridiculous like that.

The other side of data science is predictive analytics; being ahead of the game. This is where you're shifting towards machine learning algorithms. You're looking at things such as fraud, where you're trying to predict whether a transaction is fraudulent or not. Or, you're trying to figure out security applications: is this malevolent activity? But that's what it is, fundamentally. It's pattern finding within all the data, in real-time, which adds additional constraints on computational complexity.

Data science rapidly becoming something concrete, especially as it becomes a more well-

defined field. But it's definitely splitting off into those two directions of data, analyzing it and figuring out underlying trends. If there are multiple trends, maybe it's multiple elements stacking up to produce the signal you're looking at. Maybe it's not really a signal at all, and it's a bug somewhere, so you have to look at the data.

The other side is not just trying to make heads or tails of the data, but also making predictions. Which city are we going to open up in next? What are the relevant quantities? A lot of business is driven by intuition and gut feelings, and this scares a lot of people. CEOs are trying to pitch entire companies on feelings, essentially. They're trying to drive home their points on a colloquial basis. The whole field of data science is trying to turn that feeling into something a little more rigorous; trying to deliver on something that's not intuition, and finding something that you can ground yourself on. That gives your business a lot of stability, especially when there's a lot of startups and they're all thinking of great ideas, but only some of them are really as great as they believe, and most of them won't pan out.

I think data science splits into two fields, and I believe a lot of hiring companies are starting to reflect this. Data science is starting to break off into descriptive analytics and predictive analytics.

You engage a data scientist at the point when you're looking to add an incremental value. That's not going to make your business take off, it's not guaranteed. But at least it will give you something that's not solely based on a feeling.

Of the two different types of data science you articulated, do they also require different skills?

For the most part, they require a lot of the same core skills. Predictive data science requires a little more machine learning type skills, and descriptive probably requires a lot more statistical skills. But then, in predictive data analysis, you might be using a lot more random forests or neural networks – all these really cool algorithms.

Which side of data science, from your physics background, seems more intuitive with you?

I started learning programming In high school, because I wanted to play around with genetic algorithms. So that's been a long running interest. Even though I went off and did experimental physics and computational astrophysics, I've always had this background of really wanting to do machine learning. That appeals more to the predictive side than the descriptive side. Both of them have a lot of overlap. There's not a wall between the two, but you can start to see the continuum of data science. So, I think I'm far more

attracted to the predictive side. Neural networks I just think are really cool because you're essentially training artificial intelligence. You're taking these tiny artificial brains and making a decision with them. You're actually turning a whole company based on that.

What do you feel are the defining qualities of a top-notch data scientist, compared with someone who is merely good?

I think it deals with communication. I think that's the difference between the good scientists and the great. Both are going to know a lot about statistics, the techniques they can use, and how to design, implement, and execute an experiment. Those things are all important. The biggest thing, though, is that you need to be able to communicate those results. That's a lot harder than it looks.

I think the easiest thing for a graduate student to do, coming into this field, is to gloss over it, but that's the single most important thing. Most people complain that graduate students don't have a great programming background. All of their other intuitions, well designed experiments, caveated results, are sound. But I think that a lot of people believe that a programming background is not necessary.

So, maybe it is programming for a lot of people, but if you're already pretty good, then you're probably already a good programmer. The last step is just communication. People need to sense the passion inside of you. This defines the most successful people. It's the realization that you are working with other people, and for a lot of scientists, I think that's quite a shock. It really goes against this notion of romantic science.

Isaac Newton spent three years in a shack during the plague. He didn't want to get the plague and he hated talking to everyone. Granted, he was possibly autistic in some ways, but I think a lot of people follow that archetype of going back and living by themselves, and then they emerge with all of their findings. But in reality, it needs to be a much more continuous process. It needs to be a much smoother process than just coming back and reeling off a list of accomplishments. So it's always communication, but that's the easiest part to skip over.

What do you see as the promise with data science, and also the interplay between mathematics and computer science, that really speaks to you? Where does your passion lie?

We're living in a really exciting time because I think what were formerly highly theoretical principles are finally having an impact on the world. Before, I was looking at clumps and galaxies. To do that I needed to run clustering algorithms. I needed to be able to run

distributed frameworks on thousands of nodes to answer basic questions.

Now, I can do almost the same stuff, and I can tweak a learning algorithm that teaches students in the best way they can learn. There's a whole feedback system that says, "you should answer these questions, and then five minutes from now, we'll come back and repeat it, and then we'll come back a week later and repeat it again."

I joined an open source project, and that was the single best decision in all of graduate school. I learned how to code in a collaborative way.

The wonderful thing is that those algorithms, that whole pattern, is being replicated from galaxies to psychology and cognition. All of these high topics of knowledge are beginning to trickle down, and they're actually making a real impact on day-to-day interactions. There is not a single company on NASDAQ that doesn't use some aspect of this. Your Facebook Newsfeed is highly tweaked to give you everything that you think is relevant, and new content to test your preferences.

LinkedIn is using all kinds of graph networks. Square is using all these fraud detection techniques. HealthTap is fielding all of these questions, and training a computer to understand what these questions are. And there really are doctors who will be answering a lot of those medical questions.

The cool thing here is that they can take a doctor and clone him virtually. He can answer a question, and that might reduce patient time in a hospital somewhere. And when you take that power, and you multiply it by the number of patients in the whole world — it's a huge number. These are real things. We're not limited to theoretical worlds. You really can go out and have awesome effects immediately, and they're tangible. We're collecting more and more data, to the point that there are not that many aspects of life that aren't becoming data driven. So it's super exciting.

Imagine if you were able to go back to the beginning of your graduate school career, and you meet yourself coming in the corridors and you have a five minute window to speak to yourself. Would you tell yourself to do anything differently?

A lot of it would have centered on working more with people. I joined an open source project, and that was the single best decision in all of graduate school. I learned how to code in a collaborative way.

The second most important thing probably would have been communication. Every week, I would deliver a presentation on my results during the past week, and usually, it

would boil down to giving a two or three minute feedback session at the end of that. So I was already doing a lot of communication and I wouldn't have changed that.

My programming context was great; maybe I should have started that earlier and taken more formal programming classes. If you were to design the curriculum, I'd say you have to have a lot of programming. A lot of classes are like, 'go and do this assignment.' The real world is, 'go do this assignment but you only have to do this module, and someone else will do the next module. You guys need to be working collaboratively.'

They should also be doing lots of statistics, and they should be able to do it as quickly as possible. People love to talk about this Pareto Principle, where 80% of the outcomes result from 20% of the effort. The hard part is trying to figure out where that 80% line actually is, and once you realize you're at it, stop.

How can people find open source projects to participate in?

A lot of the time, they already exist. You probably already know what they are because you hear about them. The biggest thing is not to be shy about it, and not to be scared off. It took me a long time to work up the courage to actually push code back out and be able to take the criticism. No matter where you're working, there are other people working with similar problems. Just go out and search for them. If they haven't solved your specific niche problem, join the effort. It's a worthwhile process. It's really hard to convince graduate students about this, who are already overwhelmed with a lot of other things, but it is definitely the best part of those five years.

It's a little unfortunate that the primary currency of science is citations and not source code, even though that's a big infrastructure push. I think that will have to change going forward because everything is being done in a team-based context.

Your advisor is going to be pushing you for results, and my advisor said it had been years since he'd written any code. So you might not realize how important this is. But in a world that is becoming way more team-based, both in industry and science, it's super important to push everything into a team-based context.

Also, if you're in science, you're all about trying to communicate your results. One of the best ways to do that is through your open source network. They have an audience there, waiting for you, and they might be really interested. A lot of it is, 'I built this feature onto this project.' They'll go try it out and maybe they'll write a paper about it, and then you get an extra citation.

There are a lot of extra indirect effects. The direct effect is that you'll be better. The indirect effects are that there are a lot of other people who will benefit, and that will reflect very well on you.

It's a little unfortunate that the primary currency of science is citations and not source code, even though that's a big infrastructure push. I think that will have to change going forward because everything is being done in a team-based context. To do science more efficiently, it has to be that way. There's no other alternative.

ERICH OWENS Data Engineer at Facebook

The Importance of Software Engineering in Data Science

Erich sits at the intersection of data science and engineering, a role derived from his unique experiences across academia, quantitative analysis and software engineering. After training as an applied mathematician at Brown, Erich cut his teeth at Quid analysing an eclectic set of data. From Quid, he moved on to Facebook where he currently works as as data-centric software engineer — combining his deep theoretical understanding of mathematics with good software engineering skills.

He stresses the importance of coalescing different silos of knowledge and working with a business owner mindset to prioritise important pieces of work.

Please talk a little about your background and how you ended up at Facebook?

I studied mathematics and applied mathematics during college. I started with a focus on mathematics and physics at a small liberal arts college called Albion. I then transferred to Columbia University where I obtained a bachelor of science in applied mathematics.

I worked at the Stanford Linear Accelerator and the Nasa Jet Propulsion Lab doing basic research in materials science and systems engineering. I then transitioned to a Ph.D. program in applied mathematics at Brown University, but dropped out after two years with a masters degree as I couldn't see myself spending seven years on partial differential equations.

I moved out to California to work for startups. I realized that the most exciting thing for me was data science and machine learning. I spent two years at startups called Quid and Newsle and then joined Facebook four months ago as a software engineer with a bent towards machine learning and data science.

It sounds like you have this background in mathematics and you mentioned that you wanted to do more machine learning. Could you talk a little bit about how you picked data science versus the other things you could've done in industry?

I guess when you're a student at Columbia or Brown and you're looking at what kind of jobs there are, finance is a recurring theme. You go and interview for quant jobs and you

realize how there is this massive glut of smart people learning how to game arbitrage with marginal returns. In the grand scheme of things, finance just seemed fruitless.

Compare that to the Bay area, where you have people learning how to build recommender systems, teaching systems to learn and that to me is very exciting. I think personally that move seemed accessible to someone with a mathematics background. It required heavy use of high dimensional vector spaces, linear programs, kernel methods, etc., which was a language I spoke already.

On the contrary, server client protocols and the more computer science concepts were foreign to me.

You mentioned this a little bit earlier concerning your move into machine learning and data science. Now that you're at Facebook, what would you say is the value that you add as a data scientist?

In the case of Quid, they had a whole team of data analysts who were interested in having humans label training data. For them to scale, it wasn't about hiring hundreds of more people, but it was about teaching an algorithm to do what the analysts did. Their move is largely emblematic of the growth potential of Silicon Valley, where you get exponential returns by scaling hardware instead of people.

I think finding people who could play around in Python and C++ and build these learning systems was hard.

Briefly going back to your previous experience in academia, what would you say were your biggest challenges in doing research positions at SLAC or your Ph.D. program to your roles at Quid, Newsle or Facebook?

Wearing a business hat also provides a higher-level end goal which is sometimes not present in the academic environment.

Academics don't really learn to code the way engineers in the Bay Area do. You learn as an academic to hack together code to produce results for your research. There is no incentive to learn to code well or maintainably. You don't think about object orientation, functional programming or other techniques in the academic environment, which can be an impediment.

Wearing a business hat also provides a higher-level end goal which is sometimes not present in the academic environment.

How did you overcome these challenges?

I first joined Quid as a quantitative analyst and I had a very basic level of Python skills from academia. Fortunately some engineers at Quid took me under their wings and taught the basics of good software engineering.

I think when you are a student in mathematics or physics, you think vector is a vector or a matrix is a matrix, but you don't really think about how those representations tie into the computer. You don't think about sparsity, run-time, etc., which are very important in industry.

Throughout our conversations we've talked to many people about their data science background because there is such a diverse set of ways for people to get into the field. What would you have done differently through school and work given the experiences you've acquired?

I wish I plunged in more to build things, building websites or projects. When you're comfortable writing things on a whiteboard, you get scared of code. I think iterating a lot on a prototype is really empowering and lets you learn programming and languages.

I wish I had programmed more, because when I first moved to Silicon Valley, lack of coding skills was a big stumbling block.

I wish I had programmed more, because when I first moved to Silicon Valley, lack of coding skills was a big stumbling block. I think my roles at the earlier startups also demanded a lot of iteration and prototyping, which helped me learn a lot. The pressure to see results in industry made the learning process a lot quicker compared to if I were learning in school.

What would you say is the value that you bring to Facebook as a data scientist?

The value I bring is not so much as a data scientist, but as a software engineer. Although I borrow the tools of data science in terms of clustering , data analysis and classifiers, I have the ability to build a scalable full-stack system. So I am not just building stand-alone models which are pretty to look at which I'll write a paper about, but where I add real value is by incorporating that model into a scalable system.

It's really interesting that you say that because we've talked to people who say that their main value is not software engineering, but rather their quantitative skills. Of the people you've worked with, how many tend to come from the math to engineering transition and vice-versa?

Facebook has its own data science team which is full of brilliant academics. I talk to them to get advice on what features to build and what algorithms.

Having that isolated academic data science team is really useful for an engineer like me. We wouldn't be as successful without them.

I sit at the intersection of data science and engineering.

Can you talk a little bit about where data science in Facebook sits in the organizational chart or the product pipeline?

The value I bring is not so much as a data scientist, but as a software engineer.

I'm on the public content ranking team. We want to connect you to content that you may like. So in a sense we're working on a content delivery system.

In order for that to work, you really have to understand how newsfeed-ranking algorithms work and what the goal for that team is. It's one thing to rank and display your friends' content which is quite a finite problem, it's another to aggregate all content on Facebook at a given time to enable content discovery. The problem is much broader than that.

Data science at Facebook is a stand-alone organization, but I've met several data scientists who have been embedded in different groups. So the structure depends on the product. On some teams, data is used to inform product decisions, on others data is a core component.

In some ways these silos of data science remind one of Bell Labs, where you build great things and are not so worried from week to week about the details of short-term projects or metric gains.

So you're more insulated from the hard product deadlines and have more freedom to explore?

That would be my guess, but I am a software engineer. I think that would be accurate as the data science team does publish a lot of papers with the data Facebook collects.

You've been in a lot of roles from startups to Facebook, and you've surely met a lot of data scientists along the way. What would you say are some of the qualities that separate the best data scientists from the rest?

The brilliant ones I've seen at the few companies I've worked at were the ones who

could read papers, prototype and then turn it into a scalable system. I've met quite a few people who would have a great idea, but would then take forever to implement it even in Matlab.

I think strong programming skills coupled with systems-level thinking is very important. Building scalable systems may limit your ideas, but it makes them that much more powerful in terms of impact.

So I think strong programming skills coupled with systems-level thinking is very important. Building scalable systems may limit your ideas, but it makes them that much more powerful in terms of impact. At Quid, for instance, there were engineers who could build systems on their own and think theoretically. In my opinion, the combination of strong theory and the ability to implement that in a scalable manner are makes a data scientist stand out.

Are there any developments in the field of data science and machine learning that really excite you?

I like the idea of wearable computing, for instance Google Glass. Say you're in this neighborhood and you want coffee, but Glass could recommend a nearby art gallery. I like the idea of life recommendations, the idea of personal assistance, the idea of picking up on personal signals and making recommendations.

More advanced algorithms based off linear separations like support vector machines (SVMs) or deep neural networks that could learn intermediary steps or do automated feature engineering are very exciting.

Say at some point that you would like to move on; do you think that your background would facilitate an easy transition to another field?

I've thought about hypotheticals, where say in 10 years I've built a great career at Facebook and might go back to school to study quantum computers or some exciting technology at the time.

Having a strong mathematical background really emboldens you to do these things. The nature of hiring at that age is different as you would no longer be a fresh college graduate, but an experienced hire. At that point, you may also have enough experience to start your own company in an adjacent field.

How do you approach problems? What's a mathematical way of approaching data science problems and how do you use that framework to solve other problems?

I'll give you an example. When looking at time series of data, one usually opts for analyzing the entire data set which requires a large amount of memory to store, which will impede actual analysis.

Having learned mathematics and signal theory, I could use a low-pass filter and just keep a small buffer to learn the exponential moving average at any given time. You see how an analog-to-digital converter can be useful in analyzing social data. I think spotting analogous metaphors between fields is the most useful thing someone from a rigorous background can do.

Just building off analogous metaphors — simulated annealing was inspired by metallurgy. How have you found your background in mathematics useful in cross-pollinating ideas to your current role at Facebook?

When it comes to recommendation systems, people will often use singular value decomposition (SVD) to do dimensionality reduction. For me that makes sense from a mathematical background, but I've seen stumbling blocks when talking to engineers about why that concept would be useful.

I think spotting analogous metaphors between fields is the most useful thing someone from a rigorous background can do.

The ability to read a paper and understand it is also very useful. For instance there is this beautiful technique called random projections where you populate a random projection matrix using ones, zeros and minus ones, scaled by some normalization term. You can throw such a projection matrix against a high-dimensional vector and map it to a lower-dimensional space. According to the Johnson Lindenstrauss lemma, you can guarantee with high probability that the interpoint distances will be mostly consistent. It's a remarkable property because you basically scatter your data into the wind, but it's still useful with the added benefits of easier implementations and lower runtimes. It makes sense in terms of probability, but it seems really non-intuitive otherwise.

What advice or feedback would you give to people who are just starting out on their transition to the industry?

I think the most useful thing about being in college and graduate school for so many years was that I was learning for the sake of it and it was just very interesting. When I was doing applied mathematics I ironically wasn't that interested in applications. When I asked myself what I wanted during graduate school, I would say that I wanted the

autonomy to work on some really big and hard problems. That was as concrete as my career goals were.

I'm really lucky that the whole data science and machine learning industry existed when I got out of school. I worry that if I were pragmatically focused on learning certain things, I might miss more abstract concepts which have greater implications later.

So I guess I would encourage people to study what they like, but the way that worked out for me may not work for others. It's difficult to give very specific advice.

EITHON CADAG Principal Data Scientist at Ayasdi

Bridging the Chasm: From Biomedical Informatics to Data Science

After dual degrees at the University of Washington followed by a PhD in Biomedical Informatics, Eithon came to data science through a focus on machine learning applied to biology. Although initially disinterested in low level programming, Eithon came to see the powerful application areas during his research and wrote a pipeline that is still used to identify pathogenic proteins for structure crystallization. After graduating, he worked on defense projects for various US government agencies, before striking into the heart of Silicon Valley. At the time of this interview, Eithon was a manager and principal data scientist at topological machine learning company Ayasdi, where he led analytical efforts in the healthcare and pharmaceutical space. In this interview, Eithon talks about his personal journey to data science mastery, and the joy of being insatiably curious.

Can you talk a little bit about your background?

I double degreed at the University of Washington in Business and Informatics; the latter is a specialized degree that focuses on data architecture and how people interact with data and information.

I originally thought of Computer Science as a major, but took a few classes and realized I didn't want to be coding all at the time. So I opted for an option that potentially didn't involve significant programming, but did involve many things that you would typically see when working on applied technical problems.

My undergrad focus was on Ubiquitous Computing, and my undergrad capstone project was on embedded barcoding and handheld computing. I can attribute this to my first job during college, which was at Intel Research in Seattle. At the time, the Lab's stated focus was on ubiquitous computing; this means embedding computing into your environment or finding ways of using computing in ways well-integrated to the environment.

I worked on two research projects there: LabScape and PlaceLab. LabScape asked the question, "How can we embed computing systems within research laboratories to help scientists?" Basically, we'd develop studies on how scientists actually use software in the lab. PlaceLab, the second project, asked "Can we utilize WiFi devices to triangulate position and provide location-specific information to a user?" Have you ever heard of

"wardriving"? Wardriving is basically driving around town with a computer whose Wifi scanner is on. You combine that with GPS, so you know the strengths of various signals around your current location; this can be coupled with contextual information, such as stores or services nearby. Then you triangulate that information that so when you or someone else is in that area again, you can provide information on what's nearby purely from Wifi signal. Both of these were fun projects and I got to work with some really smart people at the lab.

Did you learn how to write code during these projects?

I wrote very little code initially. I wasn't a really good programmer at the time, and was not terribly attracted to coding, and for both projects I worked on the user side of the research, such as usability and user testing. I didn't become interested in programming until a later experience in biomedical science where I saw code development as another tool to accomplishing my goals.

Later as an undergraduate, I interned at Seattle Biomedical Research Institute under Peter Myler. My lab worked on infectious diseases, and my first project was to build software to identify genes. This experience got me really interested in biology and research in general.

There's a problem in biology which is pretty fundamental: you have a series of sequences, DNA, and want to identify the location of genes. At a simple level you can find stop codons, which are short sequences that suggest the end of a gene. One challenge is working out where these stop codons start, as they sometimes have many potential starting points. Over the course of a summer, I built software that used a combination of different methods to determine optimal starting position. I then used additional biological information specific to the species of parasitic genomes we were studying, to further enhance the technique.

Was that project the catalyst for you going to graduate school?

Yes, it's also the project where I realized that coding wasn't as bad as I thought, and began to appreciate the importance of computer science in modern biological research. At the time, all the computer science classes I took were taught in C, which is probably one reason I was a little turned off to the field; I wasn't great at pointers or deallocating memory. But this was a great project because it had an application. I got to see what was going on, how it had an effect, and it made me interested in learning more about software engineering as applied to science.

I had a good understanding of the computational aspects because I'd encountered a project like that very early on.

How did you choose graduate school and also how did you choose what projects you wanted to work on?

I started with a Masters. I was very interested in just learning more about computing and biology, and I knew my school actually had a degree that was specific for this kind of focus so I opted to stay there. I went to the University of Washington for my Masters in Biomedical and Health Informatics.

My focus was actually just the next step in the procedure of finding the genes. Now that we've found the genes, can we make a best guess about what it is actually doing in the biological system it operates?

I used logic and data integration to annotate the genes; biology is full of databases that have information about genes. The problem is they're so disconnected and fragmented that it's difficult for someone to make sense of it. What we had was essentially a system for collecting this information wholesale and then federating them under some uniform schema. This schema says: "Here's a gene. It translates to a protein. Here is some information associated with the gene."

I worked on this with my advisor Peter Myler and the head of my department, Peter Tarczy-Hornoch. Essentially, our solution was to treat these sources as a large database. We mapped the information of data sources in an automatic fashion to the contents of the schema. Now you can start to ask things like, "What are the functions to which this gene maps?" We used logical inference to resolve these questions. Part of this was doing some shallow NLP, allowing us to parse the gene functions and other descriptors. It worked pretty well in comparison with human scientists doing manual annotation.

That was my Masters thesis. After that I wanted to work on something a little bit different for my PhD.

While there is a lot of biology in your research, there also seems to be a strong theme of building and engineering systems. Did you continue this intermixing of biology and engineering in your PhD?

While there was a heavy component of engineering, I wasn't the only person working on the software platform on which my Masters project was built. There were a number of other folks in the lab that were working on variations of the same data integration methodology to solve different kinds of problems. In my PhD, I thought what I was doing with logic was great but there was something unsatisfying about doing something that's a series of conditional rules. I wanted something more unified.

I started to read a lot about newer statistical methods. I thought we could apply some of them to federated data for protein characterization. Data integration generated a ton of information, and statistical techniques seemed to provide a more efficient way of making sense of that information without requiring curated logic. So that's essentially what I did for my PhD. I took proteins and developed and applied a way of assigning function to them using data integration and statistical learning. Luckily, William Noble, one of the early researchers who helped pioneer machine learning techniques in biology was a professor at UW so I was able to get his help.

I took proteins and developed and applied a way of assigning function to them using data integration and statistical learning.

I focused primarily on pathogenic proteins; these are proteins in bacteria that tend to cause disease. An example would be one that facilitates the invasion of a bacterium into the host cell; another could be a protein that helps the bacteria attach to the host cell. All these functions were particularly critical for a consortium run by my advisor, the Seattle Structural Genomics Center for Infectious Disease. The Center's mandate was to characterize and crystallize as many new proteins as possible from neglected pathogens.

The overall result was an approach that gathered heterogeneous and noisy data from myriad biological sources, unified them, and then used statistical methods to determine the likeliest protein functional class. We used the method that I developed to help prioritize and classify proteins for wet lab investigation. Even now, I will get an email from my old collaborators every so often saying, "One of those proteins that you selected in your system was just crystallized, it has a structure we've never seen before." It's wonderful and gratifying to see that my work is having a direct effect on the advancement of life science.

Graduate school was a really fulfilling time for me, because I was able to spend time exploring and found the specific niche in which I was really interested. If you have a good advisor and you have good support, it's easy to do have this type of experience. The problem is that a lot of it depends on the luck of the draw. Did you pick the right advisor? Did you pick the right project? It's especially true in the sciences where if you pick the wrong project, it might not bear fruit and you feel like you lost time. So I was lucky in the sense that I had great advisors and really great research projects to work on.

It seems like although you didn't necessarily concentrate in CS in undergraduate, what you did for your master's project and also your PhD was just as extensive of an education in data integration, software engineering and machine learning algorithms. How did you learn these topics without a background in them?

I think I was fortunate at the very least. Even though I did not start with a lot of hardcore programming skills, I did have a component of them from classes an undergrad.

But I think the biggest thing that contributed to my skill set was doing graduate school. Typically, graduate school is taking a few core classes followed by going deep into something you're really interested in. As you go deep you are forced to learn the things you need to know along the way. So you end up taking extra classes, reading a lot, and meeting people who are experts in their fields because the goal is eventually to become an expert in your own field; you have to know quite a bit to advance science even just a little. At the time, as I learned more technical and analytical skills, it wasn't learning just for learning sake, but as a means to an end. For me, this perspective made the learning much more interesting.

For example, one of the things I ended up doing as an aside with some colleagues was Natural Language Processing (NLP). We ended up organizing an NLP conference and workshop in my last year of graduate school. My role there was developing a quick software system for annotating medical notes used by workshop participants, and designing the statistical technique to evaluate results.

It was a competition. We had people from across the country submit their programs and their methods to pull out information from natural language text, which is actually pretty hard. Not only did they have to extract the medication, but they had to report the route and dose for the medication, all from transcribed medical narratives. This was a nice opportunity for me to work more on both my software skills and my statistical capabilities. These are the sorts of opportunities available in research, and especially in grad school, that helped hone my skills in research execution.

Did you also enroll in graduate coursework, or were you completely self-taught?

I took quite a few CS and other classes where there was some significant programming involved. Also, in my Masters program I worked a lot on a large code base. It was a fairly significant repo, with multiple people checking in/out code. So I had to really know my way around code and know that other people were going to be reading my work; I had to write decent comments and know what's going on. Also, it was all in Java, which is a language that somewhat enforces structured overhead to begin with.

In addition, I worked very briefly for Cerner Corporation; they make one of the largest electronic medical record systems (EMR) in the US. I actually got to work on pretty cool R&D projects for their medical informatics group, and I had to keep the code well commented. Part of the exercise there was learning what is standard acceptable practice for software engineering. So that was a good opportunity to get an understanding of how engineering is done in a much larger ecosystem.

I think the nice thing about software development in general is that you don't necessarily have to focus on it to pick it up. Reading is a pretty important component, both online and in books. There's a canonical book on software engineering with regards to generating programming patterns. I read the whole book during the summer while I was working for that company. However, I don't think there's any better way than actually looking at other people's code and writing code yourself. I had some really brilliant colleagues who wrote awesome code. So getting to see what they did and learning from them was a huge benefit.

We talked to several other people and it seems like your graduate experience was very extensive. Not only did you work on these two awesome projects but also you just found time to do extra bits.

I hardly slept in my last year! You realize when you're in graduate school that as soon as you're finished, it's the real world. So I wanted to cram as much opportunity to learn as possible within the last couple of years there.

I don't think there's any better way than actually looking at other people's code and writing code yourself.

The other thing was that I got a scholarship from the Department of Defense to finish within a certain amount of time, because after that I'd get a job with the government as a civil servant. So if I didn't graduate in time I'd get a penalty of more time in the government, which was not necessarily bad, but I wanted the option to leave whenever I wanted. My Masters was two years, a standard Masters. My PhD was about three and a half. When the US government can come after you for not finishing your degree when you said you would, that's pretty good motivation to finish on time!

So immediately after graduating, did you have any thoughts about staying in academia?

Yes, I always considered it. But I had this obligation to the government to finish first. Because they paid for my last half year so I owed them the same amount in service.

One of the things that come up when people are thinking of leaving academia is that they feel isolated, whereas the draw of industry is that it's intensely collaborative. What was your experience like?

A computational biologist will often work in conjunction with other scientists. If you think about a modern biological lab, you have a Principal Investigator, a number of technicians, wet lab scientists, people doing data management and then you might have

a set of computational biologists. I think in modern biology you have a multidisciplinary team working on the same general problem where each person has their own task. One person designs experiments, another person does the data management. A different person follows up and sees if the results corroborate with models.

So in some sense it's a squad. Each person has a specialized role and has ownership. It's also very humbling because even for your small piece, you often realize there is something you didn't take into account. That's just biology: unless you have a gigantic memory capacity, you're never going to commit the entire breadth of the field to recall.

After your stint with the government, what were your thoughts on what to do next?

One thing about science PhDs is typically you have to do a postdoctorate. I liked working for the government because of the emphasis on application; what I did there was being used to make decisions. So I wanted to find a postdoctorate that would give me that similar experience while still considering government as a field of work. I ended up coming over to Lawrence Livermore National Laboratory in Livermore, California, to work as a postdoctoral scientist.

It was perfect because it gave me options. I could stay in government because it was still a government institution. However, it was also a postdoctorate so it meant I could go into academia afterward. Finally, it put me within striking distance of Silicon Valley. I felt it maximized my opportunities at the time.

After finishing your postdoctorate you had the chance then to look for academic positions, but also you were really close to Silicon Valley so there were a lot of startups and industry opportunities. So how did this influence your thought process?

I have to admit that I probably wasn't as committed to doing academia at that point. I think if someone is so close to everything that's going on in Silicon Valley you'd be remiss to not at least think about it. My postdoctorate was about to end, and the funding was lacking for this project. So there was uncertainty with regards to the likelihood of finding another project in my field there. Biodefense funding in government tends to come in waves of feast or famine.

I happened to be contacted by a recruiter out of the blue, who mentioned some mobile game companies that were really interested in finding someone to do data analysis and software engineering. So I thought, "I'm actually interested in biology and medicine, so if there's anything there that would be a better fit." She called me back later, tells

me about this company, Ayasdi, and sent me a summary of what they were doing. It looked quite interesting, and ended up interviewing and joining. I think I was the 15th employee of the company.

And you've seen Ayasdi grow from a 15-person company to now many multiples larger. You've worked in many different settings, including very applied projects, that eventually led you to the exciting world of startups, although it was never your direct intention. Do you think yourself very fortunate?

Each person has a specialized role and has ownership. It's also very humbling because even for your small piece, you often realize there is something you didn't take into account.

Let me just step back a little bit. When you pick a major in undergraduate, part of that decision is thinking about what kind of job you want after you graduate. An important part of that question is: what are the skills that you want to pick up along the way? I don't think graduate school is any different. I think one thing people in grad school have to be aware of is if they're not fully committed to academia, you have to make sure that you're building enough of a general skill set that you still have plenty of non-academic options open.

Take my grad work in data integration. I wouldn't call myself a world expert in it, but I also know it's a challenging problem that's important for people to pick up and am fortunate to have gained valuable experience in it. In many cases, it's a neglected area of expertise. So you want to be able to pick up things that you know are going to be useful.

Being able to pick that up and learn from smart people and get their advice was extremely valuable. Domain and skill don't necessarily have to be completely intertwined. You can have a domain focus (in my case it's biology and medicine), but you can still continuously pick up skills that are applicable to that domain but also have broad generalizability to others. I'm thinking about not just undergraduate but also graduate school going forward. Maybe you have a domain, but it's important to understand that there are things that aren't necessarily specific for that but are worth learning.

For me it's been tremendously worthwhile. While developing the next iteration of Ayasdi's software, we were at a little bit of a loss for how to figure out what to do with regards to the usability. I actually got to leverage undergraduate experience to help our team fix some of the early difficulties with usability.

Another example is just general analytical capabilities. The challenge in science is that statistics isn't something with which many researchers are very adept. In many

cases designing an experiment, even a computational experiment comparing methods, requires just basic statistics — enough that you can write a paper that confirms you've validated the results and you know that one method is likely better than others.

Also, I think breadth in our field is just as important in many ways as depth. Because if you're touching all this data of various size and form, it's important to have a good characterization of what you know and what you don't know. The more broadly you have that, the better, but it's also good if you have an area where you can go really deep. For me that's biology and medicine.

So with all this in mind, when you joined Ayasdi, did you ever think, "Wow, this is really different from my previous roles"? And were there places where you thought, "Wow I'm really glad I had x, y, z experiences because this feels exactly the same"?

This is a very interesting position in the sense that a lot of it is actually working directly with people. If I look back at my background it's mostly been head down working-on-numbers research. So having this component of it is quite unique for me. We are doing that heads down and work part, but there's also a very significant part that involves working with new customers and understanding what their current challenge is, and how best to convey a solution. One of the biggest challenges I had to overcome was to suppress my introversion enough that I could speak without stumbling over my words. It was a big hurdle initially, but I've had plenty of practice now and eventually got comfortable with it.

Skill-wise, one of the key conditions when you're looking at pharmaceutical data is that you have to be particularly rigorous about the statistics. This is a realm where we don't necessarily want faster machine learning. We just want traditional, well-understood methods so that we know with some level of confidence that the results make sense. Fortunately, I'd had plenty of exposure during my graduate and undergraduate studies doing statistics and designing computational experiments.

A lot of emphasis is placed on the use of more advanced methods when in many cases having just a sound foundational understanding of basic statistics is more critical. It gives you grounding to look at an experimental design and understand at a very simple level why it was design in that way. Why are these the main effects? Why are they characterizing something this way? Why did they select these number of replicates? Using more exotic machine learning is great too and is often appropriate for contemporary data problems, but at the end of the day there are some really basic things that everyone has to know.

When we say data science, most people think about just data but less about science. A lot of people who do data science in industry start with the methods themselves

and never really ask the questions such as: "What is causing the phenomena in the first place? How do I test that rigorously?"

I think one of the good things in biology is you're forced to ask those questions. Typically, even simple methods can do quite well in biology under certain circumstances. I think in some sense what we have at this company is a fairly simple method, and I think that supervised methods typically are a great way to start and a great way to start to get into more detail. But, at the same time you just have to guard against this desire to go after the most interesting method when sometimes the simplest thing will give you the best and most explainable answer. As a scientist, I always want to be able to go back and understand the underlying principles. But if you're looking at prediction, maybe that's less important.

You've worked at Ayasdi for a few years now, but you've seen the injection of data science into the general vernacular of the technology companies. How do you make sense of what people are doing with the term "data science" today? How do you think Ayasdi fits into this ecosystem?

I didn't even know this term existed until I got this position. I didn't know data was a discipline of science. I thought it was a prerequisite for science, not a study unto itself. I've heard the definition, "It's someone who's better at coding than a statistician, and someone who's better at statistics than a programmer." In some ways you can turn it on its head: it's someone who's worse at coding than a software engineer but is worse at statistics than a statistician! I'm joking of course, but that's how I feel about it sometimes since I'm well aware of my own shortcomings.

A lot of people that do this role have very interesting backgrounds. You don't have a huge majority of people coming from a specific discipline; it's mixed. When you look at something like computational biology, we're used to dealing with messy, noisy, ill-formatted data. There are quite a few people who come from a biology background that do data analytics and data science. Maybe they picked up data wrangling skills along the way to do extract-transform-load.

The other component that is also pretty critical is some kind of statistical training. At the end of the day, the term data science means you're a scientist, and you have an obligation to deliver results correctly. If you're not happy with it you go back to the drawing board. There's an important ability to understand and be able to evaluate whether or not what you've done makes sense from a statistical standpoint.

Then there is the domain expertise aspect. In many cases, we're tackling problems that are fairly difficult and that require a lot of knowledge of a particular area. Moreover,

being able to go to subject matter experts and speak the same language goes a long way to gaining credibility and trust from the person with whom you're working.

Being able to go to subject matter experts and speak the same language goes a long way to gaining credibility and trust from the person with whom you're working.

I think many of the applied science areas of study, and certainly things that involve experimentation, are where many people get a lot of broad experience. Graduate school is great for giving you that deep domain knowledge and then hopefully along the way you've picked up sufficient amounts of statistics or mathematics to speak coherently about what you've generated, as well as the technical chops to execute.

Sometimes it's just practice. For example, maybe you won't know having certain data in a specific way is a problem, unless you've seen it before and have done the repetitions to deal with it in a very fast way. If you do this enough, even a massive data set can be turned over very quickly because you've seen it before and you know exactly what to do. In some sense a lot of it is as much pure practice as it is science.

You answered the first part of my question. The second part is: "How do you see data science in general?" Do you feel like Ayasdi is doing something different from the mobile apps companies? And if so, what is that special something?

When you look at what a lot of places are doing, it's variations on the same theme. Which isn't to say that's bad or wrong; sometimes that's what you have to do and there's a big market in making that better and faster. In many cases there are tools and methods right off the shelf that one can adapt. Often, these were things that were developed just for those problems.

There's been a big focus on supervised methods. However, as data grows, there are potentially many outcomes of interest — for some problems, we may have little idea of what to train for or what the expected outcomes even should be. It's not that our current methods are deficient, but that with so much data, the number of potential questions with valuable answers grows very rapidly and cannot be enumerated by humans alone. Can we take advantage of very basic ideas in mathematics, such as distance metrics, and understand better where direct our attention?

What we do at Ayasdi is a very interesting way of looking at the large data problem. Our method applies well to high dimensional challenges or in many cases where people are completely inundated with data but have no idea where to search for potentially high impact discoveries. I can't tell you how many times I've been in a meeting where

someone says, “We have a lot of data, we know there’s something awesome in it that we can use to optimize our process/business/medicine/drug. But we don’t know where to start.”

Being able to elegantly and mathematically tackle this is going to be extremely useful as this becomes a very common problem. We’re already seeing this in many different business areas. Speaking just from my experience in biology and medicine, I see a lot of opportunity as more genomic and health information is collected and stored. There is a huge amount of value in just asking the right questions; the problem is that human ability to formulate reasonable hypotheses is finite and limited. So being able to identify and prioritize those questions in a data-driven way is going to be extremely important going forward.

What do you see coming out of the pipeline in the next three to five years in medicine and computational biology that you’re excited by, and that wouldn’t be possible without the new data tools and techniques being developed?

Medicine is a very interesting field. It’s a field where you need a lot of domain expertise to be really proficient. Take a physician for example who has to go through many years of school, sometimes more than a PhD, to be really good at what they do. How do they do that? They don’t do that by reading formulas or theorems. The biggest and most important component of their training is going to the hospital and seeing patients.

That’s how medicine is. But I think as we move forward it’s going to be critical to inject a lot of data there. To capture data, understand what’s going on, understanding how different practices affect results and outcomes. That’s not even touching the genomic aspect and personalized medicine. There’s so much information, there’s so much variability in how patients get treated across the board. How do we help make this more uniform?

Optimizing patient outcomes is a very interesting problem because it’s something that’s always been there and surprisingly, we do all these things with data but that’s still something that everyone wants to solve. I think that’s one area in medicine where where data science can help to make a positive and lasting impact.

I think that one of the things you mentioned very early on that doesn’t get talked about much is just how much work it takes to be really good at something. When you first started working, you worked very late into the night, most nights doing analysis. What drove you, and drives you as a data scientist today?

I think probably the biggest thing was curiosity. That’s actually a huge component. When I’m up late and trying to figure out a problem, my personal goal is to better understand

what is driving whatever phenomena I am observing. I just want to find out. I've had this plenty of times here when working at Ayasdi, where I find myself facing a problem that is just driving me bonkers because I want to understand it better. Maybe I found this pattern and that pattern, and they all point to the same thing, but the outcome is the inverse of what I thought! So I want to dig deeper and know why. I think like any scientist the thing that drives me the most and really compels me to work late into the night until the sun comes up is curiosity.

If you could catch yourself in graduate school walking out of a lab at 3 a.m. and had a chance to speak to yourself, what would you have told yourself? How would you have lived life differently? Or would you have chosen anything different?

I think like any scientist the thing that drives me the most and really compels me to work late into the night until the sun comes up is curiosity.

In terms of general direction, I probably wouldn't change very much. I love biology and medicine, and the work is tremendously fulfilling. For the benefit of people who are interested in the field and doing data science as a career though, I would definitely say take as many statistics courses as possible. If anything, I would have told myself, "Hey! I know you don't want to take that statistics genetics course because you have a completely full load, but take it anyways because you'll end up using that information in your career at some point." I end up having to look back at scattered notes or books because I didn't take that statistical genetics course. Take more statistics courses, and take more math courses; though focus on statistics more than anything.

GEORGE ROUMELIOTIS

Senior Data Scientist & Data Innovation Leader at Intuit

How to Develop Data Science Skills

George arrived on the fabled Stanford campus in the early 90s as a postdoc from Australia. After a couple of years doing research in theoretical and computational plasma astrophysics, the beating drum of the tech boom of 90s drew George into the unconstrained world of dotcom startups.

Undeterred by the Dot Com Crash, George went on to found JRG Software, which provided scheduling software for the food and beverage industry. His time as an entrepreneur proved to be an invaluable experience in making him a holistic Data Scientist.

Today, he is a Senior Data Scientist & Data Innovation Leader at Intuit, a leading provider of personal finance and tax software. His interview touches on the minutiae of the hard technical skills, but also the macro and people skills which combine to make a holistic Data Scientist.

George has since taken a role as a Distinguished Data Scientist at Walmart.

Could you start off by telling us a little bit about yourself?

I'm originally from Australia, where I completed an undergraduate degree in applied mathematics at the University of Sydney, and began a Ph.D. in physics which I completed in the U.S. My focus was on theoretical and computational plasma astrophysics. More specifically, I investigated the physics of solar flares. I was a Senior Research Scientist at Stanford for several years before I realized that getting a tenured academic position was going to be difficult -- there were so few spots in my sub-discipline. Around the same time, I started getting very interested in business applications of applied mathematics, and eventually I decided to make the leap out of academia and into business.

In the course of my research, I had developed Bayesian image processing techniques for astronomical images, which led me into machine learning, which in turn led me into online learning. Now, those were the days when everybody was starting a company, so I decided to join the party. I co-founded Dynaptics with a few business partners, raising about $5M. This start-up pioneered the development of adaptive learning systems to optimize online advertising. A non-technical marketing manager could "release" multiple advertisements into the system, and the system would learn in real-time

which site visitor should be exposed to which advertisement in order to continuously optimize the revenue stream. As the site visitor behavior changed over time, the system would automatically adapt. Those were very exciting days, and our customers included MSN, eBay, and Cisco. Not so exciting was the Dot Com Crash in 2001, when the doors for additional funding slammed shut overnight. It was like nuclear winter for venture capital. We could not scale down our operations fast enough, so we had to shutter the company and sell off the technology and intellectual property.

Undeterred, I went on to found another start-up, JRG Software, with another set of business partners, this time raising about $10M. That start-up was in a completely different domain, namely factory scheduling for the food and beverage industry. The problem we solved was to enable factories to rapidly adapt to changing demand without holding a lot of inventory. One of our early customers was General Mills, which still uses our system to schedule all West Coast production of Cheerios! The business challenge was how to penetrate the headquarters of large companies like General Mills where SAP was firmly entrenched. We were eventually acquired by a public company that added our scheduling system to their product line.

At that point, my wife said something along the lines of, "*Perhaps you should look at doing something other than a start-up next,*" and I eventually arrived at Intuit as one of its first Data Scientists.

You were at Intuit before people started calling themselves data scientists, right?

That's right. And it's been a fascinating journey.

Along with the rest of the world, over the past five years Intuit has dramatically evolved its thinking regarding the applications of big data and advanced analytics. Five years ago, the focus was entirely on marketing optimization. Then, starting about three years ago, the scope increased to include improving the user experience by analyzing how users are interacting with our products. Now, the focus is squarely on leveraging big data and advanced analytics to create new products that solve important problems for our customers. Our unique aim is to deliver "Big Data for the Little Guy, which empowers individuals and small businesses by allowing them to benefit from the power of their own data as well as the collective wisdom of millions of fellow Intuit customers. This means that small businesses now have access to insights that were once only available to big, multi-million dollar companies, and enables consumers to put their own data back to work for them.

You worked at Intuit before there was a lot of hype and discussion about this term "data science". As someone who's been in this field for awhile, what are the myths

and what are the truths when people talk about big data and data science?

You might have heard the joke, "What is a Data Scientist?" The punchline is, "A Data Scientist is a data analyst, who just happens to live in California." I think the hype will go away, but Data Science will be a permanent feature of the business landscape. Data Science is its own unique discipline, combining elements of applied mathematics, computer science, business consulting, and, increasingly, new product development. I consider a good Data Scientist to be like a Swiss army knife, competent across all these areas, with deep expertise in one or two of them.

I consider a good Data Scientist to be like a Swiss army knife, competent across all these areas, with deep expertise in one or two of them.

More specifically, the technical table stakes for a Data Scientist are advanced statistics, machine learning, SQL and Hadoop, and a mainstream programming language like Java. So there's a combination of applied mathematics and computer science. But of equal importance are business consulting skills. These are often overlooked, or added as an afterthought, but they are critical. Business consulting skills can be the difference between a Data Scientist and a Data "Gopher".

A Data Gopher is someone who responds to incoming requests for analyzing this or that, but who never has a seat at the table when the business decisions are being made. On the other hand, a Data Scientist with business consulting skills is like a senior McKinsey consultant, who can translate fluently between business and technical domains, and who is a trusted advisor to business leaders. Those are highly non-trivial skills.

When you talked about skills required in data science, you talked about three things: classical statistics or machine learning, computer science and business consulting skills. What suggestions would you make to someone looking to build those skills?

In terms of database skills, it is essential to feel completely comfortable with SQL and Hadoop. If you are still on campus, for goodness sakes take advantage of that by signing up for relevant basic courses that include a major project component.

In terms of programming skills, learning R is very important. It is kind of ugly, but it is the lingua franca. Personally, I would stay away from proprietary, commercial statistical programming languages. You know the ones I'm talking about. And certainly you need to learn a mainstream programming language like Java or C++. Learning a mainstream scripting language like Python or Perl also comes in handy.

If I had to assign a weight to help someone prioritize all this learning, it would look something like this:

SQL	40%
Hadoop	30%
R	15%
Mainstream programming language	10%
Mainstream scripting language	5%

Don't become a Data Scientist who has never operated as much as a lemonade stand.

In terms of acquiring business skills, you have to get creative. At Stanford there was a fabulous entrepreneurship course tailored to engineers and scientists. Simply listening to lots of entrepreneurs tell their story is very helpful. Subscribe to *The Harvard Business Review*. Talk your way into a challenging internship that presents you with an open-ended problem. Above all, just start an online business. It doesn't need to be the next Google. Give yourself the challenge of starting with $100 and seeing how much you can make it grow in a month. That can be quite eye-opening. Don't become a Data Scientist who has never operated as much as a lemonade stand.

You have a very unconventional experience in that you left your postdoctorate position to found companies. Not only did transition from a postdoctorate to a business environment, but you jumped into the deep end and decided to start your own company. What types of thinking did you feel like you benefited from during your experience in academia and what are the things that you felt were hindrances to you when you entered the business world?

Having the foundation of applied mathematics was extremely useful, because then I could pick up other math-based bodies of knowledge very easily. On the Ph.D. side, I mainly learned persistence.

What certainly didn't help me, and what I had to unlearn, was how academics present their results to others. As academics, we're trained to take an axiomatic approach. "*Here at the start of the presentation are my axioms, and here in the middle are the detailed steps that I took, and then here at the very end are my results.*" But if you do that in a business meeting, and you hand out copies of your slides beforehand, you'll observe that the first thing the business leaders do is flip to the back of the deck to see your conclusions. They just don't care about the detailed reasoning, because that is your job, not theirs. I have found it much more effective to start with "the bottom line up front" and then show the thought process if there are questions. This is a very different mindset from academia.

Also, in academia you get kudos and endorphins from doing something novel. But in business it's all about the efficiency with which the company can transform money into more money. So a Data Scientist needs to resist the impulse to solve problems *ab initio*, or to spend time going from the 80% percent solution to the 90% solution. That effort sometimes doesn't make much business sense. You've got to think about allocating your time as though you were the owner of the business.

Intuit is a very data-centric and financial-centric company.You didn't start out in a business context, so what framework do you use to evaluate the success of potential ideas at Intuit?

Don't fall in love with your own ideas. Market feedback is the only thing that matters.

The way I look at business has definitely evolved, especially from being at Intuit. I've learned to take a hypothesis-driven, experimental approach to developing solutions to business problems. We should all feel passionate about the problems we are solving, but we must not fall in love with our solutions. We design experiments to let the customers choose between Solution A and Solution B, rather than that choice being made by "the loudest voice in the room." That's a mistake I made in start-ups, and one that I saw a lot of other people make as well. We were all convinced that of course we knew what the market wanted, and we proceeded to spend a lot of time building it. The way I work now, and the way I would have advised my younger self to work, is to create minimalist prototypes and test them out on real customers. Don't fall in love with your own ideas. Market feedback is the only thing that matters. You've got to do experiments, and you've got to be ruthless about changing your ideas based on the results of those experiments.

We've interviewed people who have recently made the transition to data science, but as someone who's seen the growth and development of younger data scientists, what are some of the mistakes often made by younger hires?

First, you have to proactively build relationships with your non-technical colleagues. Data Scientists are often by temperament introverts, but if you want to be effective and successful, you need to step outside that comfort zone. Email a non-technical colleague you've never met, and ask them to lunch. Make it your responsibility to form such relationships before you need them.

Next, practice viewing the world in terms of business processes. What's a business process? It's a foreign concept to many new Data Scientists coming directly from academia. A business process encompasses the people, systems and steps involved in a business activity. Generally speaking, a Data Science project has the goal of improving

some existing business process. The truth is, it's really difficult to change a business process.

For example, it took me a long time to grasp that improving the efficiency of a business process might actually be perceived as threatening to someone's job, and the natural reaction of that person might be to consciously or unconsciously undermine any progress. So you have to develop deep empathy for the people involved in business processes, and create solutions that help those people transition to higher-value work. That sounds like a lot of responsibility for a Data Scientist, but if you don't think about things like that, your ideas might never be implemented in the real world.

Beyond these three attributes, what does it take to be a successful data scientist, in your opinion?

It took me a long time to grasp that improving the efficiency of a business process might actually be perceived as threatening to someone's job, and the natural reaction of that person might be to consciously or unconsciously undermine any progress.

A successful Data Scientist changes the world around them. It comes down to mindset. One mindset is that your responsibilities are to analyze a situation, construct a solution, and then pass along that solution to others for implementation. But that is a recipe for frustration for anyone who is interested in moving the needle in the real world. A better mindset is to think of yourself as the business owner who is responsible for changing how the business works. That's a whole different mindset, one of taking ownership for how your ideas are going to be implemented and measured. The additional skill you need is influence without power. How do you influence others to move forward with your recommendations when they don't report to you?

How do you influence others without power?

It is not easy, that's for sure.

As I said earlier, it starts with being proactive in forming relationships with your non-technical colleagues, because people want to work with those they know and like.

It is also important to go for small wins before trying to hit the ball out of the park. Small wins prove that you are a reliable partner.

And you have to make the connection between your recommendations and the bottom

line. Yes, that's often very hard. There are usually many links in the chain between your work as a Data Scientist and the outcome for the business. But nobody else will do that analysis if you don't. It goes back to having the mindset of the business owner.

When you're looking for data scientists, do you feel there is a necessity for having any form of senior academic credentials? A lot of the data scientists we're seeing now have a Ph.D. background, but do you think this trend will continue into the future?

Back in the day, when relational databases were brand new to the world, the folks who were most comfortable with that technology were at IBM Research. It is not surprising that the first relational database experts in industry had Ph.Ds, but over time the barrier to entry has obviously been reduced. Data Science might be like that. Maybe. On the other hand, Data Science might be more like brain surgery than SQL. I think it is too early to tell. The well-rounded Data Scientist is competent in applied mathematics, computer science and business. Such people don't exactly grow on trees.

It is not surprising that the first relational database experts in industry had Ph.Ds, but over time the barrier to entry has obviously been reduced. Data Science might be like that.

What distinguishes a data scientist from someone who works as a data analyst in a traditional business intelligence role, or a statistician with programming knowledge? How deep do these distinctions go?

Statisticians might be steeped in mathematical tools for inference and prediction, but that alone is not going to make them an effective Data Scientist. They also need to be completely self-sufficient in extracting and manipulating large data sets that are usually found in legacy systems, and which are often a noisy mess. They need SQL, NoSQL, and programming skills to do that. And even if they have all the programming skills, but they don't have superb consultative skills, they will have very limited influence. I think a Data Scientist is a very different animal from a statistician. But that's just my opinion. I'm not interested in getting into a religious argument about what constitutes a "real" Data Scientist.

You were the CTO of the first startup you founded and that seems to suggest that in addition to being an excellent physicist, you're also quite skilled at building systems that provided software solutions built off your interests in image processing or scheduling. Were those programming skills something you picked up during the course of your graduate degree or was that more born out of need?

I found that software engineering courses at Stanford taught you how to write a program, but they did not necessarily teach you how to work in teams, or with diverse systems that you have to integrate. And they did not teach you how to build, deploy and maintain complex software. All that relates to project management and people skills which are usually not addressed in computer science programs. So I learned those skills via trial and error. Lots of error!

I acquired software engineering skills by actually building the Version 1 solutions at the two startups I co-founded. I think it's very hard for Data Scientists to work effectively with software engineers if they haven't done any software engineering themselves. I don't think a Data Scientist necessarily needs to be a production software engineer, which is a different mindset yet again. But basic fluency — knowing how to write, document and test code, and how to create components that are used in larger systems, that's important.

Where do you think data science is headed?

I don't think a Data Scientist needs to be a production software engineer, which is a different mindset yet again. But basic fluency ... that's important.

I think we are going to see an explosion of both consumer and enterprise products that are made possible by Data Science — that is, by the creative melding of big data and advanced analytics. To achieve that, some Data Scientists will need to become product designers, adding the skill of "design thinking" to their toolbox. Deep customer empathy, rapid iterative prototyping, and in-market experimentation will be essential to this emerging sub-type of Data Scientist. Or maybe we'll need a new name. Data Product Designer, anyone?

DIANE WU Data Scientist at Palantir

The Interplay Between Science, Engineering and Data Science

While studying computer science at Simon Fraser University, in Canada, Diane became interested in biology. After graduating, she began a PhD in genetics at Stanford University, where she also dabbled in courses in computer science and machine learning. Diane's background in genetics naturally prepared her to working with large volumes of data, leading her to realize that the work she engaged in at Stanford naturally belonged in the realm of data science.

After graduating, Diane became part of the Insight Data Science Fellowship program where, as a Fellowship project, she built recipe searching site that used clustering to organize recipes by ingredients.

When we interviewed Diane, she was a data scientist at Palantir. She has since started a new role as a Senior Data Scientist at MetaMind.

To start off with, how did you get started in data science?

I did my undergrad in computer science and became interested in biological problems, so I transitioned to bioinformatics and did a PhD in (Computational) Genetics at Stanford. While I was there, I started taking classes in the CS department, in part out of interest due to my CS background, in part because I loved interdisciplinary approaches, but also because these were rumored to be the most challenging classes.

I took Machine Learning with Andrew Ng, Probabilistic Graphical Models with Daphne Koller, Data Visualization with Jeff Heer, and Mining Massive Data Sets with Jure Leskovec. I took these out of interest and because I thought they would be applicable to what I was doing; sequencing and essentially going through terabytes of DNA sequences to make sense of them. I was doing a lot of time series clustering, predictive modeling, and building Bayesian models. I took these courses because I thought they would be relevant to my research, but what I didn't realize through the entire process was that I was basically doing data science in biology.

When I finished my PhD and decided I didn't want to be in academia, I stumbled across the Insight Data Science program, specifically designed for helping PhDs transition into

industry. Through this program, I realized that most of the training through my PhD was essentially just data science. So the transition for me was very natural. I'm doing a lot of the same things, just not thinking about cells or biology! However, the same tools and challenges apply.

So after bridging this gap, where are you as a data scientist right now?

I work as a data scientist at Palantir — a company that builds a platform that helps integrate data for our customers from the multiple disparate databases that they have, and makes associations and inferences from these data. We work with customers from the financial sector, the medical space, government and local law enforcement. One of my jobs as a data scientist at Palantir is to help create value out of their data and identify a human-computer symbiotic approach to machine learning.

Data science itself is a very strange term. It's an umbrella term.

Given that you're working with these large institutions, what is the scale of the problems you're tackling?

There's a wide range. Some of our customers have hundreds of terabytes of data and some have a few megabytes. Some customers require a streaming solution while others want a static model based off all the information in their databases. The number of databases we work with can also vary between one to many dozens.

Having worked as a data scientist for a while, what would you say are the main responsibilities and goals of data scientists at Palantir?

Data science itself is a very strange term. It's an umbrella term. In some companies and in some roles, being a data scientist means to be a software engineer, building machine learning models in the back end. In this role, your success is very measurable--it is usually the accuracy or precision/recall of your model performance. At other companies or in other roles, being a data scientist means that you're an analyst working with engineers to help them determine what features to build and how users are interacting with them. In this role, your success is less measurable, and it is up to you to find the right questions to answer and then to try to make impact with that answer.

At Palantir, we work with customers from a diverse number of sectors, with a wide spectrum of problems that we solve by deploying our platforms against their data. One of our core company missions is to pick incredibly difficult problems at institutions where we would provide the most value, and put our full force into solving these problems.

Sometimes, this means developing new capabilities in the platform. Sometimes these capabilities require data science techniques (machine learning, statistics, mathematical modeling), and that's where we come in. I'm on the machine learning team at Palantir, and we're dedicated to enabling customer data science needs via our products. To this end, we work closely with customers to help them scope their problems and turn an often poorly defined, qualitative problem into a quantitative one. The process involves identifying an actionable goal or desired insight, evaluating the form, scale, reliability and availability of the data, and building custom machine learning algorithms to solve the problem. And then we iterate. Always iterate.

Some requests we get involve translating from qualitative problems to quantitative ones (identifying good proxy metrics to reach the right conclusion), statistics (doing the calculation on the data), and communication (presenting the data in a digestible manner). In most cases, however, our customers are requesting a predictive analytics approach to a specific type of problem. They present a very difficult problem where a predictive modeling component may be needed. Fraud detection is one of those problems, for example. It is clear that a computational algorithm could aid fraud detection by identifying patterns and outliers, but the problem is complex enough that it will likely always involve a strong human component. In such cases, it is not clear how we should break up the tasks between the human and the computer. One of Palantir's core values is human-computer-symbiosis: let the computer do what it does best (crunch models, calculate metrics, etc.) and let the humans do what they do best (interpret patterns and meaning, make accountable decisions, especially with respect to the rights and well-being of other humans). One of the overarching goals of our team is to figure out what an ideal predictive analytical solution should look like and where on the spectrum it should lie.

Finally, we also do data science internally, and often want to use product metrics to inform business decisions. Engineers like to build cool things. It's not intuitive to them to think about things in a scientific way. I think that's one of the reasons books on lean product development are so popular. It's because these are not intuitive concepts for engineers. The role of a data scientist is to do the stuff that is a pain for engineers (but fun for us), and help engineers make more data driven product development decisions.

It sounds like data scientists are evangelizing the scientific method to engineers!

In a way, I guess that's true. It's intuitive to me to think in the scientific mindset because I've been trained as a scientist for the past 4 years. It's very natural for a scientist to ask why, to dive into a problem, scope the hypothesis landscape and then perform tests. However, scientific thinking is a double-edged sword, and is in some ways the opposite of the engineer mentality. Scientists ask why something is the way it is before reaching a conclusion, while engineers execute on assumptions and watch to see if things break.

One of the hardest things in recruiting for data scientists is to find candidates who have the right balance of both scientific and engineering mentality. Almost always, with real world problems, there is no time to ask why and figure everything out before executing, and you often have to act with incomplete knowledge. However, engineering without data science is like building a bridge without ever fail testing it. There is a delicate balance to be struck.

What are some challenges and some of the things you've found easy in making the transition from PhD to Data Science?

The reason why programs like Insight have been successful is because PhDs have been trained with a quantitative method of thinking. They're also prone to ask "why" and "how" rather than "what". I think that most PhDs understand the presence of errors, and how to reduce a complex problem to a smaller problem with a quantifiable solution.

> *One of Palantir's core values is human-computer-symbiosis: let the computer do what it does best (crunch models, calculate metrics, etc.) and let the humans do what they do best (interpret patterns and meaning, make accountable decisions, especially with respect to the rights and well-being of other humans).*

On the other hand, PhDs are often stereotyped to ask "why" too often and are sometimes caricatured to have their heads in the clouds. So if I find a PhD who is also a hacker, then it is the best of both worlds. Indeed, some of the most effective data scientists I've seen have been PhDs who worked on a number of side coding projects during their academic career.

The challenge for a lot of people is the ability to apply these insights into value. Not all interesting problems can produce insights, and not all interesting insights can inspire action that causes change.

Did you have any challenge in communicating your value as a data scientist?

What I have learned in working with many different customers is that when people request data science, they really just want magic. They want you to use all the data to predict everything. When they approach data science, they often don't actually know what they want.

That's the thing about being a data scientist in this time. It's so new and sort of overhyped, that most people just know they want in on the excitement but don't know how. They want things, but they have no true idea about what they want.

Part of the job is really use-case discovery. It's not always about crunching the right algorithm. It's about asking the right questions and framing the questions for yourself. And once you do that, the problems tend not to be statistically or algorithmically hard.

On the other hand, there are people who think it's overhyped and want you to prove that data science is worth their investment.

So in your experience, what distinguishes the best data scientists from the rest?

That's a very good question.

There are statisticians and there are computer scientists and designers. And then, there are people who are very good at all of these things. The reason why this role — data scientist — was created, and the reason why it's a little bit undefined, is that it requires that you're good at many different things. You have to think about problems, both as an engineer and also as a statistician. You have to know what tests are right, how to approach the problem, how to engineer the solution and how to sift through large datasets.

And then afterwards, you have to present your findings in a clear way. This might require you to create visualizations. Having an understanding of graphic theory and the language of visualization is useful. This ties into communication because as a data scientist you're communicating with someone who doesn't have a ton of time to analyze data. They look at the figure and want to be able to extract meaning from it in a few minutes.

Finding someone who's a good engineer and a good communicator is incredibly difficult. You don't need to be the best at everything, but some people who are great communicators need to learn how to be great engineers and vice versa.

In academia, there's a focus on open-ended problems. How have you made the transition to industry where there's an environment to deliver on prompt deadlines?

I think in an ideal world there should be a fusion of the two. In academia, it behooves one to work with deadlines; most PhD students would probably tell you that if it weren't for publication deadlines and the fear of being scooped, we might never publish. Open-ended problems need to be scoped also, and often a 20% solution will get you 80% towards your goal. In industry, sometimes people can get too "hacky" and deliver v1 solutions all the time, and that can be bad too. Sometimes it's good to step back and try our hand at some crazy ideas. I think that's the inspiration behind company internal hackathons and why they're so popular in the tech industry.

What skills beyond what you've already mentioned (hypothesis testing, communication) would you recommend to someone interested in data science?

As a preface, I think the skills you need to learn largely depend on what you want to do.

I would put this into three categories:

1. **Predictive Modeling:** here, algorithms and some complex mathematical modeling are required. Visualizations are probably not as heavily emphasized.
2. **Business Intelligence:** here you engage frequently with SQL and some scripting, but you don't need great skills in computations and algorithms.
3. **This is a spot in the middle:** this is more science-y and R&D. Here you want to ask much deeper questions about user behavior. You want to model user interactions and apply computational algorithms to gain business insights. This is a mesh between two extremes. You need some computational background, and some aspects of communication, etc.

But ultimately, to answer this question requires you to think about what type of job you want, and realizing that you can't be qualified for everything. You have to pick your best shot and hone your skills there.

Building off of that, what have you found to be useful in building those skills and understanding which position you want to pursue?

Talking to people is important. I don't mean that in the way of networking, but in the way of understanding what people are looking for. Insight Data Science brought me a lot in this direction.

It's about asking the right questions and framing the questions for yourself. And once you do that, the problems tend not to be statistically or algorithmically hard.

Looking at folks who have moved into data science, I've noticed that the Coursera course by Andrew Ng has been very popular. This ties into the general skill of being driven enough to simply pick up books and start learning. A lot of aspiring data scientists also play around with some Kaggle competitions to get their hands on real data and practice their engineering and analytical skills.

In fact, most data scientists I know are self-motivated, they've taught themselves the relevant tools and skills to help them manipulate and understand data. In my opinion, it doesn't take that long to learn these skills. So if you pick up these things after work, I think you can take advantage of the large demand right now in data science.

Kevin Novak, another data scientist we spoke with at Uber, believes that we're at the tip of the tip of the iceberg when it comes to data science. Do you agree with

that? And if so, what are the exciting and promising things on the horizon of data science?

I agree with that. I think that data science is largely undefined. Being a data scientist in this time is exciting because you have a lot of potential to define what data science is for the next 10 years. What's exciting is being able to explore this frontier. You're also learning a great deal about very different fields intersecting with each other. I really like this position because I'm learning so much and I'm not just honing one skill.

I'll predict that in 10 years we'll use more defined terms than data science because people will realize what it is that they're looking for (analysts vs. predictive modelers).

Are there any final thoughts or parting feedback you'd give to someone just getting into the field right now?

Don't be afraid!

March forward and learn what you have to learn. Many people who come into data science are overwhelmed. They look at the list of "requirements" and think that because they're not a wizard at engineering, or a statistician and a visualizer, that they're not qualified.

I think they shouldn't underestimate themselves. I think you should approach things in the T-Shaped model, where you accumulate a great deal of breadth and a concentration in one skill that gives you depth.

So be confident and pick up skills; you'll be surprised at how much value you can add immediately.

JACE KOHLMEIER

Dean of Data Science at Khan Academy

From High Frequency Trading to Powering Personalized Education

As an undergraduate student in Kansas, Jace wasn't exposed to the ins-and-outs of high-powered finance. Little did he know that within a few years of graduating, he'd be working at one of the largest hedge funds in the world.

After receiving degrees in math and computer science, Jace went off to Princeton for a PhD in theoretical computer science. There, he was lured into the startup space at the height of the 1999 bubble, where he learned that he preferred the entrepreneurial process to proving theorems.

After leaving Princeton, Jace joined Citadel, where he worked for 7 years before starting his own trading firm.

His time in finance incubated an idea of "high frequency education." After leaving the world of trading, Jace looked towards a different field: education. After hearing Salman Khan's TED talk, Jace felt compelled by Sal's vision and joined the Khan Academy as the Dean of Data Science.

You're currently the Dean of Data Science at Khan Academy. What is your background and experience up to this point?

I was a math and computer science dual major in undergraduate school, and I also did a couple of internships in the software industry. I then entered a PhD program in computer science at Princeton, where I intended to focus on theoretical computer science.

But that was about 1999 and it was at the height of the Dot-Com hysteria. At Princeton, I met some people involved in the startup space and decided to take academic leave after only one semester. I went to an incubator in New York to start a software company. The venture ultimately didn't turn out to be commercially successful, but after experiencing the entrepreneurial process, I learned something tremendously valuable — I learned that entrepreneurship was more appealing to me than working on theorems for five to six years.

What were some of the key differences between your computer science graduate studies and the accelerator? What did you find that changed your mind about industry versus academia?

The question should be: "Why the heck did I ever think grad school was right for me?" Because in retrospect, grad school, or specifically PhD studies in grad school, were wrong for me in just about every way. I had always been pretty commercial. I got my first job as a programmer when I was fifteen. I've always loved markets. When I was a boy, I scoured my monthly baseball card price guides with great enthusiasm. And while I do love math, my experience in grad school was fairly solitary. I found it to be mostly time spent with my head in a book, sitting in a library or literally in a windowless basement trying to prove math theorems.

It was lonely and slow and felt kind of devoid of any exciting risk. My experience in New York was just the opposite. It appealed to my commercial senses and I loved the intensity and time frame of trying to get software shipped or a product developed before the money ran out. I enjoyed the camaraderie and the teamwork, and basically just felt far more excited and alive.

So what did you do with these realizations after the incubator in New York?

For years when I was working in finance, I'd been fostering an idea about "high frequency education," of using rapid feedback loops to test educational content and pedagogy.

What I chose to do was to go back and finish my Master's degree at Princeton and then look for something more commercial. Living in New York had given me opportunity to meet some people in quantitative finance, which up until then I didn't understand or even know existed. As a kid from Kansas, I had no idea that people were combining math and computer science in awesome ways and applying it to finance.

This is something that really opened my eyes. Rather than trying to work at the very depths of one particular subject, like computational complexity, quantitative finance seemed like the combination of all three of my main interests--the market, computer science, and math. The chance to take three of my loves and package them perfectly into a professionally rewarding space was a no-brainer for me.

I went back to Princeton and through on-campus recruiting, landed a job at Citadel — one of the world's largest hedge funds.

Was that your first full-time job? What was that like at Citadel?

I joined Citadel as my first full time job out of college. I had a good experience there and after about 2-3 years, a few other people and I started a new business within Citadel that centered on high frequency trading. We traded a range of securities via sophisticated statistical models and algorithms. That internal group turned out to be very successful, and after 7 years at Citadel, I was able to start another trading firm with a partner.

Given that Citadel was your first full-time role after college, how did you approach learning new things in quantitative finance?

The coding side of it pervades all of this work. The faster you can code, the faster you can implement ideas. If you have a good sense of building systems, you can scale what started out as a research project into something operational.

My job at Citadel was probably the first time that I really needed and wanted to learn about how to build empirical models. That was not something that I had ever really studied in school. Maybe in passing I came across a regression model in a statistics class, but basically, I was starting from scratch at Citadel. My approach — which may not have been optimal — was to start reading books. Sadly, there were not the great online resources that exist today, which is what I would now advise. I read books and I tried to pick the brains people around me that were doing the quality of work that I wanted to do. I hung off of every word that they would tell me or give me in terms of mentorship, which I sought. There's no doubt that my most essential lessons, for both hard and soft skills, were learned from my mentors.

So how did you eventually end up turning to education?

After co-founding a trading firm, I decided to seek another challenge and at that point, I was looking for something different. Education was something that I had always been interested in and it also runs in my blood. My father was a high school teacher; my sister was a high school teacher and is now a professor of education. For years when I was working in finance, I'd been fostering an idea about "high frequency education," of using rapid feedback loops to test educational content and pedagogy.

So that was a pet vision of mine; how we could port key ideas I had used within high frequency trading to education? As I was looking for what I wanted to do next, I explored various options. I even volunteered in a Chicago South Side school and took the exam to become certified as a teacher in Illinois. But when I came across Sal Khan's TED talk in 2011 where he was describing a system of exercises and videos to optimize education, I was interested right away.

Let's talk about the questions that you attack at Khan Academy. What are the algorithms and problems like? How do you measure improvements to learning from Khan Academy's platform?

Oftentimes, we know the thing that we want to measure and so we can apply statistical techniques to try to measure it very efficiently. Occasionally, we ask the user a question from a distribution that we control, but there are costs to that because the question we want to ask may not be what the individual wants to learn. So combining knowledge from information theory or graphical modelling, we can treat the answers as evidence and try to elicit the most information for a minimum cost to the user. This approach requires that you are fairly conversant with quantitative techniques.

The coding side of it pervades all of this work. The faster you can code, the faster you can implement ideas. If you have a good sense of building systems, you can scale what started out as a research project into something operational. We can work much faster if we are data scientists *and* engineers. We can build algorithms and models right into the product, but that obviously requires that we be competent in the product's engineering.

Of the people whom have applied for a data science role at Khan, what really stands out to you as being fundamentally core skills and what are the skills which can be learned on the job?

As we built our team, I became increasingly skeptical of finding the perfect data science "unicorn" — someone that is world class in all of these categories.

The hardest thing to teach on the job is a strong quantitative aptitude. Most people who apply would probably rank highly in that regard. They've been studying mathematics since they were 5 or 6 years old, and continuing through college, so it's taken a long time for them to build up their knowledge base. These quantitative skills are definitely the hardest to pick up on-the-fly given the amount of time that needs to be invested, but I also don't think that picking it up on the job is impossible.

We have developers or other quantitative scientists who may not think of themselves as being experts in machine learning, but who are clearly very technically minded and sharp. So, I don't mean to say it's impossible to learn quantitative aspects on the job, but simply it is the hardest thing to pick up on the fly.

Somewhat similar is coding experience, which is a productivity gauge. If you're 30% slower at coding, you have less time to focus on other aspects and your productivity will go down. We look for *fluency* in coding.

The hardest thing to pick up when interviewing candidates is a person's aptitude for experiment design (model design) and how these experimental outcomes will impact your organization. We've experimented with bringing people in for on-site interviews involving projects. Another initiative we have tried is to put candidates through collaborative exercises as well.

As we built our team, I became increasingly skeptical of finding the perfect data science "unicorn"— someone that is world class in all of these categories. By definition there are very few people who are world class in even one of those dimensions, and they are in extraordinarily high demand. So you really want to put together a team similar to the way a GM puts together a professional basketball or baseball team. There are fundamental skills that all players share, but the GM puts together a team of complementary members that specialize in position or area. More and more, that's the way I think about building a data science team.

Given that the background of the ideal data scientist you're describing implies a deep expertise within some fields, do you find that the people who are predominantly trained in these areas come from advanced degrees?

At Khan Academy, there are no medals or ceremonies if we publish a beautiful research paper. What we really want to do is deliver demonstrable value to people trying to learn. A critical skill for a data scientist is knowing how one's work fits within a team, and where your team sits within the concentric circles of an organization.

I think of team as an ensemble of specializations. I have seen a PhD experience be both a benefit or a drawback in a couple of ways. I've seen a PhD be a benefit when a candidate or employee has really learned to independently find their path through a nebulously defined problem, or to be able to craft experiments to get to a result that's going to meaningfully answer pertinent questions.

For some people, it's very clear that their PhD led them to develop that skill. On the other hand, some people's PhD experiences left their pragmatism atrophied. At Khan Academy, there are no medals or ceremonies if we publish a beautiful research paper. What we really want to do is deliver demonstrable value to people trying to learn. A critical skill for a data scientist is knowing how one's work fits within a team, and where your team sits within the concentric circles of an organization. In some cases that can be a skill that may have atrophied for someone who comes from a doctoral setting.

You've mentioned the importance of programming several times now — for aspiring data scientists who come from a strong quantitative research field, some might not

have spent so much time with software engineering. What are some ways for them to increase their programming skills?

In my opinion, to be a great data scientist, you must be a great (or at least a very productive) programmer. That doesn't mean that you have to be a savant in computer science, it just means that you have to be fluent with code and experienced in building real systems.

The great thing today — which wasn't available in my day — is you can get involved with open source projects and get very specific feedback from great developers. This is a tremendous resource and opportunity for people who want to improve their programming skills.

What I would suggest for someone who's looking to build skills in those areas is, number one, you just have to write code and you have to write a lot of it. There will always be differences between a first year programmer, a fifth year programmer, and a tenth year programmer, at least for people who spent those years practicing the right way. The hack to get better faster is to get lots of good feedback. And the best way to get feedback is to find great developers to work with who will give you code reviews.

The great thing today — which wasn't available in my day — is you can get involved with open source projects and get very specific feedback from great developers. This is a tremendous resource and opportunity for people who want to improve their programming skills. So write a lot of code, and make sure you're getting code reviews from quality programmers.

On the process of implementing the machine learning algorithms — how do you learn more data science on the job?

There is not a steady rate at which you learn new techniques and employ them; it definitely comes in waves. When I made the transition into this new domain of education and internet-generated data, I went through a period of needing to learn new modeling techniques. I wasn't familiar with probabilistic graphical models; that wasn't something that I had used in high frequency trading.

Once I got past that initial learning curve, learning came very much in waves. There will be a very concrete and motivating need or goal. For example, we're very focused on delivering value to the users and so any new foray into a new modeling technique is usually driven by that goal. Often we will have the necessary knowledge at hand. If not, we'll take the time to learn what we need to know.

Another idea that we often hear in our interviews is the importance of communication and how to cultivate more interpersonal skills. Either through their research or just natural personality, some people might be introverts. What advice do you have on how to manage communication when work relies on collaboration and teamwork?

One of the greatest things that anyone ever did for me professionally was during my time at Citadel. My boss's boss, came to me and said, "Hey, we think you have potential but there's something that you really need to work on, and it's your communication skills." They put me in "communications training", which was both useful and hilarious.

That's a great question and one that I can relate to profoundly. I would describe myself as a pretty hard-core introvert and it's something that I have continuously had to work on in my career and continue to work on to this day. One of the greatest things that anyone ever did for me professionally was during my time at Citadel. My boss's boss, came to me and said, "Hey, we think you have potential but there's something that you really need to work on, and it's your communication skills." They put me in "communications training", which was both useful and hilarious.

I was videotaped role-playing various business scenarios, which felt totally bizarre. I thought, "I'm a quant, this is ridiculous!" Then I watched the tape and I was appalled by looking at my body language and hearing my verbal mannerisms. I'm still working on this today. Despite how silly it seems, I totally recommend that my fellow introverts try the videotape technique. Andrew Ng recently shared a great post on how he used a similar technique to become a better teacher and presenter.

Another important development for me was partnering with someone who was very much an extrovert. That helped me in two ways. It gave me an exemplar for dealing with other people effectively. And, it taught me that it's OK to lean on a trusted partner at times to handle the extrovert work, while I remained focused on my strengths.

So those are a couple of strategies that people can use. Number one, get yourself feedback — possibly through videotaping — and conscientiously work on your communication. Second, seek partners with extroverted tendencies that can complement your more introverted tendencies, and build extra close relationships with those people.

That's fantastic advice. Switching gears to diving into your work, what's a day like in the life of a data scientist at Khan?

It's fast paced. The Khan Academy engineering team, in general, is very focused on

shipping and iterating quickly. We try to ship code everyday. So fitting into that, what I focus on and what my immediate team focuses on is what we call "learning gain," and we try to take a very pragmatic approach to the work. We're not looking to just produce research, we don't pat ourselves on the back at the end of the day for writing a nice report or making a pretty graph. What we really want to do, which sounds grandiose, is to change the lives of our users through more effective or increased learning through Khan Academy. That is what we measure ourselves by and the questions that we ask must relate in some way to that goal, and hopefully, as directly as possible, to the main question, "What are we doing to improve learning through Khan Academy?"

The data that we deal with is almost entirely generated from user activities on the website. Occasionally, there are some complimentary external data sets, like geography, but it's almost entirely user activity, which forms both their practice and their assessment, so to speak.

When you're doing analysis, you're not really writing code anyway. You are using existing machine learning libraries and you're basically an intelligent matchmaker between the data and the appropriate model.

A good day is a lot of code writing because it's the most direct way that we build value. Then, as a team lead I also need to make sure our team is in sync with the organization. I learned some hard lessons during my first couple of years at Khan on that front: A) make sure the product design is amenable to the research requirements, and B) promote ideas for doing experimental research that may not occur to others. For example, we might read something in the science of learning, like there's strong evidence or justification for this particular style of learning, and we think it could be studied in this particular way. If we communicate this to the engineering team, they might be able to add this into the product, and we would be able to measure and build off of that.

So a good day is mostly writing code, checking in on the real time results from our A/B testing dashboard and then doing the interesting work of talking to other teams to understand how they are making their decisions, and what can we do to help them. We stay focused on the product because at the end of the day, that's what the users touch. That's our ultimate goal. So if we're not changing the product and changing it in a way that delivers better outcomes to users, then we're not doing our job.

Lets talk about the future. How do you see the foray of computational statistics into computer science? Do you see that data science will also become commoditized? How do you think data science will evolve?

I agree there's been a heavier commoditization of big data services, which has reduced the need for people with data infrastructure backgrounds, though I would argue that those skills are still very valuable and some of the infrastructure tools are still relatively immature. I think there's a lot of improvement to be made there, but you can see it coming. So that leads to more emphasis on the next layer of actually analyzing the data.

If I have data, and there's a standard suite of off-the-shelf models, how do I combine those techniques and know which technique I should use? I think the emphasis will stay in that area for a long time. When you're doing analysis, you're not really writing code anyway. You are using existing machine learning libraries and you're basically an intelligent matchmaker between the data and the appropriate model. I don't see that machines are good matchmakers, because this involves knowing which tools to apply in which contexts. It's a fairly high level process, which for now requires human intuition. So I think we will need people with those skill sets for a while.

You have done a lot of interesting work in high frequency trading, and now you're working in the education space. What excites you about your job and what are some future prospects that excite you?

Still, one thing that surprises me is how much harder the problems I work on at Khan Academy are compared to the problems I worked on in high frequency trading.

At Khan Academy, we give our new hires a couple of sci-fi books. One of them is *The Diamond Age* by Neal Stephenson. In the book, Nell is a young girl who comes upon a sophisticated guided learning tool designed in the form of a book, and it teaches her skills and knowledge in an interactive way. The author even accounts for the presence of human-backed interaction with people called "ractors," that can bring some essential humanity to Nell's experience. It's an amazing vision that seems tantalizingly within reach. None of the material seems far-fetched in the age of iPads and other educational technologies.

Still, one thing that surprises me is how much harder the problems I work on at Khan Academy are compared to the problems I worked on in high frequency trading. Everyone assumes that developing a model that consistently generates money in the financial world has to be the hardest thing to do, but I think it's a different breed of difficulty. There's a reason that what we now call data science originated in finance, because finding signals and knowing what to optimize is so ingrained in the profit and loss (P&L) of financial trading. The signals and optimization objectives are so much less well defined and quantifiable in the education space, which makes it hard from a value creation perspective. So the objective function is not as well defined. In education, we

run into a host of considerations for defining the objective function. Should we optimize for educational breadth or depth? Is the answer to that question the same for everyone? What about their emotional state? How do I incentivize people to stay engaged in learning? There are more human aspects which blur a well-defined objective, but which also make the work very challenging.

I've immensely enjoyed my work at Khan Academy, and I think my dream of achieving technology that facilitates completely personalized education is both realistic and epically ambitious. *That's* exciting.

JOE BLITZSTEIN

Professor of the Practice of Statistics at Harvard

Teaching Data Science and Storytelling

Joe Blitzstein is a Professor of the Practice of Statistics at Harvard, moving to Harvard after obtaining his PhD in Mathematics from Stanford. He co-taught the initial offering of Harvard's inaugural Data Science class, designed around the Data Science Process. He also teaches Harvard's popular introduction to probability class as well as other statistics courses. Joe regularly emphasizes intuition and storytelling in his instruction. He tweets at @stat110.

Joe Blitzstein is a co-author of the textbook Introduction to Probability.

How did you get interested in statistics?

I was a math major as an undergrad at Caltech. Caltech doesn't have a statistics or data science department, and there are also very few statistics courses there. I went to Stanford for graduate school in math. It hadn't really occurred to me at the time, but Stanford has a ton of statistics and data science-type opportunities.

I was working on probability for my PhD thesis because I really love it, but I decided that it's better if you can have your cake and eat it too. With statistics, you can actually do cool math and also feel that you're analyzing interesting data and doing something useful for the real world. It still has nice mathematical structure and lots of elegant thinking. You can also feel more useful, whereas math, itself, tends to get more and more abstract and disconnected from reality as you progress. Statistics, though, is rooted in the real world and real data, and data science is a version of statistics.

Can you describe for our readers what Harvard's Data Science course is like and what is the philosophy behind it?

It's a course that I created with Hanspeter Pfister, who is a visualization professor in the Computer Science department. Our goal was to give an accessible introduction to the entire data science process.

We defined that process as a journey, starting from formulating a research question

and gathering data. Then, you clean the data, so there's some data wrangling. There is exploratory data analysis, which involves looking for problems, biases, weird outliers, or strange anomalies in the data, as well as trying to get a sense of some possible conjectures you could formulate.

Then, it goes a little bit into modeling. We took a Bayesian approach to that. There are full courses on Bayesian data analysis, and this was just a short introduction. Then, there's communicating and visualizing the results.

The sequence of steps is not linear — you iterate between those steps in a non-linear way. We defined it as the data science process, and we wanted to introduce that process through examples. To go into detail, that process would have taken six courses, but we wanted to put them together into one introductory course on how to think like a data scientist. The course needed to include applications that are of current interest like predicting elections, movie and restaurant ratings, and network analysis, rather than using a lot of canned, stale data sets that no one has cared about in the last 50 years.

So, we wanted interesting data, but that's not enough. We wanted interesting data but we also wanted to ask relevant questions about the data.

Why is important for data scientists to understand the data science process instead of just going through the work?

With statistics, you can actually do cool math and also feel that you're analyzing interesting data and doing something useful for the real world.

I think it's important in whatever you do to have a sense of direction, instead of just aimlessly trying things. You want some sense of where things are going. I'm not saying it's not useful to just grab data and hack around with it. You can learn from doing that, but in terms of doing something that will have long-term scientific value, I think that depend on relevant research questions.

Much of statistics is about distinguishing signal from noise, distinguishing valid from invalid signals, so-called "discoveries". You need to look for patterns, but you can't just assume that whatever pattern you find is real. You have to perform some validation, and if you cannot communicate the results in the end, it's not worth much either.

All of these ingredients are crucial. Different people can specialize in different parts of the process. No one can be a complete expert at every step, but data scientists in industry are working in teams. To be effective in teamwork, you have to understand some basics

of what your teammates are doing. You need to be able to give them feedback, and you need to be able to understand their feedback about what you're doing. You have to see how the various pieces fit together into the overall process.

How did you get interested in data science and teaching this data science class?

It's probably a combination of a lot of factors. I noticed more and more possible data science ideas and applications ever since the Netflix prize and Nate Silver. The combination of so many datasets that were never available before made me really interested, for my own sake, as well as for teaching it to students. I felt some concern that students might not have the right kind of CS training to be able to participate in all these opportunities. So, I wanted to play a role in fixing that.

Your data science class was very popular this year. Did you expect this level of popularity? How many students ended up enrolling in the class?

I had guessed there would be 100 or 150 students (which would already be a very large course), but we ended up with more than twice that many; we ended up having 350 or so enrollments. We tried to keep the prerequisites reasonable, but it did require at least some very basic background in Stat and CS. We didn't want to limit enrollment or do a lottery, so we tried to send the message that this was going to be a hard class. You're going to do a lot of work, but you'll learn a lot. That was the idea, but I didn't expect it to have that much demand.

Why do you think there was such a large demand for the class?

It's hard to know. I think there are some students who took Stat 110 and wanted to have a follow-up, even though the material is different. In Stat 110, we do probability and it's a fairly mathematical course, but we're not analyzing data. In a data science course, we're not doing math, but we *are* analyzing data. I see Stat 110 and the Data Science course as complementary, in that we are emphasizing stories and a certain way of thinking about the world in both of them.

So, it's the applied analog, and I have a huge number of Stat 110 students who were interested in going further. Then, Hanspeter had a lot of students interested in his visualization course. The visualization itself is great, but it's very limited if you don't actually know how to analyze the data. So, the whole theme of big data attracts a lot of interest.

You mentioned emphasizing stories when you were comparing Stat 110 and the

Data Science course. I want to extend that question: What is the role of storytelling, communication, and visualization in data science?

I think they're incredibly important parts of it. Anyone with a basic level of CS can scrape a big data set and start computing things. And anyone with the right statistics background, if presented with a clear data set, can start running some regression in a mechanical way. I think there's a real art to getting interpretable results and then communicating those results, especially in the age of big data where you have thousands of variables. In the old days of regression, you might have two predictors, and it's a lot easier to see what's going on. Now, we have thousands of variables and some very complicated models, and it becomes very difficult to see what's going on.

I think communication includes communicating with yourself too! You are trying to make sense of the data in a way that human beings can understand. If you attend conferences, it's generally hard to remember anything from the majority of a presentation. Presenters tend to rush through their slides and try to show a lot of results, but are they really explaining what the story is?

Much of statistics is about distinguishing signal from noise, distinguishing valid from invalid signals, so-called "discoveries".

So, if statisticians (or anyone) are falling to communicate why their results are important and are failing to explain those results in an interpretable way, that's just a lot less exciting. Visualization definitely plays an important role in that case. A picture is worth a thousand words. Sometimes instead of staring at a huge table of numbers, a few graphs can give you much more intuitive information.

Do you have any advice for data scientists or people in the industry who may want to become better communicators? What kind of philosophy would you like to impart to make them care more about the storytelling and communication part of data science? Why is the teaching part of data science so important?

I think it's an important part of clarity of thinking. As a data scientist, you're going to need to collaborate with many different types of people with many different backgrounds. You have to be able to put yourself in their shoes and explain things in terms of what they're interested in and what their background is. In many cases, when you can't explain something clearly, it's a sign that you haven't thought it through fully yourself. So, teaching and learning go together. Learning to explain something to someone in an interpretable way makes it a lot clearer in your own understanding.

In terms of concrete advice on developing these communication skills, I think of it in terms of something like the golden rule, which I call the conditional golden rule: try to present the idea in a way that you would have appreciated seeing it presented. It's conditional because you have to adjust for the fact that as a data scientist who's been immersed in a project for months or years, you have to step back and realize that the person you're talking to may have never even heard of what you're doing. They don't know any details about the data. They don't know your notation, and they may not even know statistics.

In many cases, when you can't explain something clearly, it's a sign that you haven't thought it through fully yourself.

Also, read some of the classic design books by Edward Tufte (he's a famous example), *The Visual Display of Quantitative Information*. Try to find and follow good examples.

What's your opinion on his book and his philosophy on visualizing information?

I really like his books. In a sense, he's a victim of his own fame, in that these books are so popular that it's almost a visualization bible. So naturally, there's going to be a backlash of people asking, "what gives him the right to say what you can or can't do?" I wouldn't take everything he says religiously, but these are important things to think about. Clear communication is incredibly important.

What are your favorite philosophies about visualization? What is your favorite piece of knowledge from this book, and what is your best advice for visualizing quantitative information?

I think the best advice is just to think hard about what you want your audience to take away from the visualization. It's sad to think of how many talks I've been to, presentations on all kinds of subjects, where the speaker will make ridiculous mistakes, like not labeling their axes or having things so small that the audience can't see what is going on.

Sometimes, presenters want to show some kind of comparison, but the things they're trying to compare are on separate slides. Graphs are effective in showing something changing over time or a comparison between things, and it is more about relative information than absolute information most of the time. You want to make it as easy as possible to see those comparisons. Avoid something that looks really fancy but distracts attention from the fundamental comparison you're trying to display.

Can you tell our readers more about your story behind the conditional golden rule?

There were two course reviews about Stat 110 that went well together. One of them said

I designed the course to the credo that it should be taught in the way I myself would like to take it as a student, which is the golden rule. Then, the other one, which is a counterpoint to that, said that the homework only induced pain, not learning. The joke is, that if you combine those two things, it implies that I'm a masochist.

Obviously, I'm trying to induce learning, not pain, but it does require a lot of hard work to learn all these things. I try to make as many resources available as possible, in terms of having great Teaching Fellows, having lots of office hour times, and having large amounts of practice problems. It's just like if you were practicing a sport or a musical instrument. It's something that you need to practice, practice, practice. Just doing a few homework problems a week is not going to be enough.

It's like learning a whole new language. Language courses tend to meet every day, and you have to go to labs. There are tons of things going on, but statistics and data science are new languages, too. They should be approached in the same way. You have to do the math and CS as well as learning grammar and syntax. You just have to immerse yourself in the learning process.

For my fellow students and me, we're very fortunate to be in this environment where our only duty is to learn. But there are many data scientists out there who feel like they're missing some knowledge and are trying hard to fill the gap. My question is in reaction to those data scientists. What's the best way to keep on learning after university?

It's a dangerous way of thinking — that until you know X, Y, Z and W, you're not going to be able to do data science.

I noticed that's a trap that people fall into, thinking, "I'm perpetually feeling unprepared." It's a dangerous way of thinking — that until you know X, Y, Z and W, you're not going to be able to do data science. Once you start learning this thing, you realize there are four other things you need to learn. Then, you try to learn those things, and you realize you don't have this, this, and this.

You do need some basic foundation in statistics and CS skills, but both statistics and computer science are enormous fields that are also rapidly evolving. So, you need durable concepts. Right now, for people that want to do data science, I highly recommend learning R and Python. But in 10 or 20 years, who knows what the main languages will be?

It's a mistake to think, "why am I learning R now? R won't be used in 20 years." Well, first of all, R might still be used in 20 years, but even if it isn't, there's going to be a need for the thinking that produced R. The people who create the successors to R will have

probably grown up using R. So, they're still going to have that frame of reference.

You want the skills that are language-independent. You need fundamental ways of thinking about uncertainty and communicating those thoughts in a way that is not that dependent on any particular programming language. It's definitely important to have that kind of foundation, but keep in mind that it's hopeless for anyone to actually know all the relevant parts of statistics and CS, even for some small portion of data science. It's not feasible for anyone, but it doesn't mean that you can't make useful contributions.

You have to be energetic and work really hard, but not get discouraged just because you don't know everything.

In fact, I think it's a good idea to continue learning something new every day. The way you can learn something, and really remember it, is by using it in your work. Instead of saying, "I need to study these five books so that I will know enough to become a data scientist," it should be about getting a basic level and foundation. Then, start immersing yourself in a real, applied problem. You will realize what types of methods you need. Then, go and study the books and papers that are relevant for that. You will understand them so much better because they're in the context of a problem that you care about.

You have to be energetic and work really hard, but not get discouraged just because you don't know everything. And just because you don't know everything, it doesn't mean you can't contribute useful things while gradually expanding your understanding and knowledge.

To strengthen one's understanding in a concept, would you also recommend teaching that concept to other people (stemming back to your philosophy on storytelling and communication)?

Yes. I think that's a great way of checking your own understanding. It's a lot of fun. You're helping someone. You have to think about the important things to emphasize, the common misconceptions, etc. Think back to when you first learned the concept, the obstacles and conceptual roadblocks that you had to get past, and the most important things to emphasize. That is very useful for everyone.

What are the parallels between being a data scientist and being an educator?

Communication and feedback. If you're just lecturing to a class and not paying attention to see if the students are actually understanding, that's a pretty stupid way to teach. There's a story of a professor who got a really poor teaching evaluations, and the

evaluations said his lectures were very unclear. He said, "My lectures aren't unclear. The students just don't understand."

Communication is a two-way street and you have to pay attention in various ways, through feedback, watching people's expressions, trying to get people to speak up and feel comfortable asking questions. Do whatever you can when you're teaching to assess what people understand and what they don't. A lot of that information stays the same from year to year.

That's the reason why every week in Stat 110, I ask the teaching fellows for the most common mistakes from the homework. I can clarify those things or they can be clarified in the sections for that year. Those things tend to stay fairly constant from year to year, too. I don't have a formal data set, but I am trying to gather as much information as I can about what the students understand and what they do not.

Data science is like that, too. You don't compute something without getting feedback on whether it's working or not. You're communicating messages to people, but you need feedback on whether or not that message is getting across.

This is a very important idea in software development, too, with continuous deployment and instant feedback and quick iterations. It's nice connecting data science and software engineering principles. As a data scientist, you're always getting feedback and trying to improve.

I think that's extremely important. That's another mistake I've noticed, the tendency in applied problems where new students just want to fit one model and be done with it. But, the world is too complicated. There are too many challenges with data. We know the saying, "All models are wrong, but some models are useful."

It's not realistic to expect that the first model you come up with will actually work well, but if it takes too long to figure out how to fit that model and run the computations on some massive data set, you may feel that you need to move on.

That's very unsatisfying. What you need to do, first of all, is get comfortable on Python so that you can fit the models very quickly. If you have a large data set, fit it on a subset first so you can quickly get models and better intuition. You have to iterate and build something better.

You have to manage your time so that you can actually go through a whole series of models and get feedback on which one is actually working through measures of fit or predictive capabilities. Even just explaining or communicating with someone else to try

to get a sense of whether the model is actually doing something useful and what aspects need to be improved, is very good.

Building off of that, what other mistakes do you see made, or what other things do you think data scientists should improve on?

Statistics is just a hard subject and there are all kinds of mistakes made. I think the biggest one tends to be not thinking enough about sampling. Where did the data come from? What do you want to assume about possible selection bias or other forms of bias in the data? If there's a systematic bias, no matter how large the data set, it still has to be accounted for. It doesn't wash away in the limit of large data. You can't necessarily think of the data as being an objective, unbiased portrait of reality.

Another thing is trying to be very clear, at all times, about what the goal is. What are you trying to estimate? What are you trying to predict?

How can university education better prepare students for opportunities in data science? What skills are students missing in general?

There are very few data science courses in universities currently. There's a large number of statistics and CS courses that are closely related, but there are not many courses that integrate statistics and CS. There are also not many courses on communication, visualization, and storytelling.

I think the problem is that there are not a lot of clear paths to follow. There aren't data science majors generally, and even the statistics major is small or nonexistent in most schools. It's a recent trend that statistics even exists as a major.

One purpose of a major is not just what degree shows up on your diploma. It's also about providing tracks, having some coherent path through the material. For data science, you definitely want a strong combination of both statistics and CS, and different schools certainly vary in how much relevant material they have in each of those. I think very few have developed a road map, a coherent ordering, and a scheduling of how to get the requisite knowledge.

There are going to be a lot of readers whose schools don't really offer these kinds of opportunities. How do you feel undergraduate or graduate students can get the required data science knowledge when universities don't offer these specialized tracks and courses?

It may or may not be a formal online course, but there's a massive amount of excellent

online material. I really can't curate it right now. That's something I'd like to do at some point, to try and recommend the most useful online materials, and try to recommend a curriculum. I know others have tried to do that.

But now, in this era of big data, it's even more important to understand experimental design.

There are a lot of great books, but like I said, you don't just want to read a stack of books. Maybe you can read or study a couple of books or do a couple of online courses, but just try to start doing something like Kaggle competitions at Kaggle.com. They have very interesting data sets and problems, often about predicting some quantity. Try out one of these competitions or find some data you're interested in and just get a feel for it. You can look through different resources about regression models and machine learning. Look at different materials and then try it out on the data set you're working with. You'll get a lot more intuition about which methods work well for which problems.

A lot of those things are very hard to teach in a course anyway. Even if universities offered more data courses, many of these things are best learned through hands-on experience in internships, in competitions, or with just playing around on your own with some data.

Do you have any funny or interesting anecdotes to share with our readers?

I read an article in *Wired* recently, about the importance of A/B testing at Google and various tech companies, and I thought it was nice that they're calling attention to that. The thing that was funny and sad about it was that the article made it sound like they had never heard of the history of experimental design and randomized controlled experiments in statistics, which goes back to R.A. Fisher in the early 20th century.

That is 100 years of history on how to efficiently design experiments. For example, if there are many variables you're interested in, it's really inefficient to design an experiment in which you only change one thing at a time. You want to have a full factorial design where you're changing different variables at the same time. It's much more efficient, and there's a lot of good theory and application for randomized experiments. Some people consider randomized experimentation as one of the biggest breakthroughs in medicine in the 20th century.

The article was talking about A/B testing, which is just a trendy word for randomization, where you have one treatment group and one control group. The article went on to speculate, is it even conceivable that we might be able to A/B test the offline world, too?

So, I thought it was funny that, apparently, they never heard of a randomized

experimentation, but also, it was a reminder that data scientists need to be familiar with traditional statistical concepts, such as experimental design and sampling theory, in order to be able to be much more efficient in how they deal with data.

Experimental design was one of the main topics in statistics in early 20th century, and at some point, it started having a reputation of being a bit old-fashioned. It was only old-fashioned because it was one of the founding topics on which statistics was created. But now, in this era of big data, it's even more important to understand experimental design.

So, it's coming back but with new challenges related to the new types of data sets that we have. It's very interesting to see old and new ideas come together.

Lastly, what advice would you give for undergraduate and graduate students who are interested in going into data science?

Whenever possible, ask fundamental questions like, "Who cares?"

I just recommend getting a mixture of math, statistics, and CS background, to build a strong foundation. Then, concurrently, immerse yourself in as many real world applications as you can, remembering that depth is often better than breadth. Immerse yourself into challenging problems that you can hopefully integrate with your coursework, so that you see some ideas and how they are or are not relevant for particular types of data science questions.

When you're learning, constantly question and constantly be critical. Whenever possible, ask fundamental questions like, "Who cares?" Constantly think about the motivation. Why is that relevant? Why is this data set interesting? What questions could we hope to answer? When you're trying out different statistical methods, don't just use it like an off-the-shelf, black box type of thing where you just spit out results. Question! Do those results make sense? How do you assess whether the method you're using is working well, or how do you know if it's actually working better than random guessing or using a complicated method? How do you know that it's better? In what way is it better? Is it better than some simple, naïve thing you can try? Constantly try things and compare them. Question whether or not the results make sense.

JOHN FOREMAN Chief Scientist at MailChimp

Data Science is not a Kaggle Competition

As an undergraduate math major, John thought that he was going to be a pure mathematician. A few experiences working as a programmer, combined with a talk with his advisor, pushed him instead into the world of applied math.

After a sojourn in academia through MIT's Operations Research PhD program, John realized that a long-term career in industry would be more interested and fulfilling.

John held a series of jobs in business intelligence at various consulting companies, before taking on the Chief Scientist role at MailChimp, a fast-growing, completely bootstrapped, email startup based in Atlanta Georgia that boasts over 7 million users.

He is also the author of the book "Data Smart," which presents an overview of machine learning techniques, as explained through spreadsheets.

Can you start off by talking about your book, "*Data Smart*"? How did the motivation for writing it come about, and what type of audience is it for?

I felt like there were a lot of business analysts and middle managers in the enterprise world who were not familiar with "data science" as a practice and set of techniques. These folks still lived in a world of "business intelligence" or "business analytics" from a decade ago, and I wanted to bring them up to speed on current methods (for example, ensemble AI models built on transactional data, data mining in graphs, forecasting with error bounds). I wanted to get these enterprise readers up to speed, so I needed to find a language and teaching approach that they'd understand.

A lot of data science books that were being introduced at the time required learning both R and techniques at the same time. With a lot of those books, rather than actually learning the techniques, you just loaded the AI package or the data mining package.

Instead, I wanted to write a book that introduced these concepts step-by-step with a tool the reader knew, and then, once they got it, slowly push them into a programming mode. So in *Data Smart*, I explained the gamut of data science techniques by using spreadsheets. Spreadsheets are kind of like a functional programming language and GUI in one, and they're actually pretty good for step-by-step model building.

The last chapter is an introduction to R, and it harkens back to previous chapters now that the kernels of understanding had been planted. For example, if you're doing an *exponential smoothing forecast* (which I cover in my book), you should not be doing all these steps every time. You should be doing it on the shoulders of the giants who've written the Ph.D. theses you're using and just open their package.

Ultimately, people who want to know each little detail of how a boosted tree model works or how modularity maximization works seem to love the book. Programmers who are used to relying on black-box libraries, functions, etc. aren't the biggest fans.

Given your interest in opening up black boxes to examine the nitty-gritty of different techniques, did you ever want to write your thesis on a new statistical or machine learning technique?

I started at MIT wanting a Ph.D., but in my first year of graduate work I had the opportunity to do some applied work on Dell's supply chain, and it showed me that my passion lie outside of academia.

You see, my advisor was really interested in publishing results. Although we came to Dell trying to understand how to help the business generate revenue — which I enjoyed — that wasn't our ultimate goal. The problem with consulting when you have ulterior academic purposes is that the goal of academic publishing is counter to the goal of helping a business, because in order to publish, you need something academically new to say. But if it's a new technique, it is often not maintainable by the business once the academics leave.

That was a good experience for me, because I realized that I'm not an academic despite the fact that I like technical things. Rather, I'm an analytics professional who enjoys tailoring technical approaches to business settings where the solutions are sometimes complex but often simple, depending not on my needs as a data scientist but on the business's needs or the customer's needs.

That ability to think simply and "edit" models is something I just published an article on. One thing I reference in the article is a paper from 1993 by Robert Holte titled "*Very Simple Classification Rules Perform Well on Most Commonly Used Datasets.*" His basic premise is that simple decision rules – a single rule that splits the data on one feature — are pretty effective compared to more complex models, like a CART model. That makes sense since oftentimes in naturally occurring data sets within the business, you have a couple of features that are good, and everything else is just icing on the cake.

One of the things Holte says in the paper is your model complexity has to be justified,

and that really grabbed me.

It made me think, especially in a business context, what it means to justify your complexity. Part of that is the additional expense of keeping the model running versus revenue. Part of that is the poor sap you're saddling with keeping this thing running. And something that people don't often consider is the likelihood of abandonment.

Your model complexity has to be justified, and that really grabbed me.

Once you move on, whoever gets saddled with it might find some organizational reason or anecdotal evidence to ignore it. They may not even tell you or get back to you. Are you going to stick around and babysit all your models? How do you hand them to someone if they're complex?

So to come back around to getting a Ph.D., for me, it was this desire to go out and use data to serve a business, and to do that using both complex and simple approaches that pushed me to leave graduate school early and join the workaday world. I don't regret it.

It seems like the academics are trying to do the most complex models, and the business decision makers are thinking it may not be all that helpful, that 80% of the way there is good for us already.

Can you talk more about your background? What were you doing before your PhD?

My dad is an English professor so I thought I was going to do English. Slowly, I realized I was pretty good at math. In my undergrad, I studied pure math. I really liked abstract algebra, and I thought I was going to be a pure math guy. My advisor sat me down and said, "You're alright. You'll probably go to grad school in a top 10 program, but you're really not going to amount to much in the math community." I felt at the time that it was pretty harsh, but it was true. I couldn't compare myself to other people doing pure math.

The way math works is a lot of people toy with little results for a long time, and suddenly there are huge jumps from certain people. I would never be that individual who would push the mathematical fossil record forward into a new era. I would be a guy that toys with smaller results. So it came down to a question of passion: how passionate was I about pure math?

At the time, I was also doing research for another math professor on knot tying. I got

paid as part of this research group to write code that would take two 3D models of knots and join them into a compound knot without crossing over other sections of the knots and forming a new knot type. It was crazy specific, but I learned a lot about Unix and programming. I wrote code to do simulated annealing in C. I was getting all sorts of memory leaks, and I had to do a lot of stuff in the command line with data sets.

I didn't know what that was at the time. I thought it was just math research that involved code, but I liked it. It turned out to be my most valuable experience as an undergraduate. After all, what would a data scientist do without piping in Unix?

What did you end up doing once you graduated?

I did a couple of internships at the NSA over the summers, and I loved the applied, problem-focused environment. When I did my first summer internship, it was all math students hopped up on stories of Bletchley Park, etc. Lots of energy. It was great, but then I did another internship there and they put me in a regular office with regular employees that had been there for a long time. And that's what ultimately scared me away.

I remember talking to one guy who had a picture of a golf course above his computer. He said, "That's what I'm doing next year when I retire, playing golf." Everyone was tired, and everyone was burned out. I figured that a government job wouldn't be exciting for long, so I began to look at other applied analytics opportunities.

So in graduate school I chose to study operations research where math was applied to optimization modeling. I went to MIT in their Operations Research Center which is an interdepartmental program between engineering, stats, math and business. It was cool because you could take business classes alongside highly technical classes. I got a kick doing MBA case studies because it was so foreign to a math class. No proofs!

I thought the OR program was awesome, so I knew that career-wise I was headed in the right direction. When I did my graduate research for Dell and was able to use the OR concepts in a consulting framework, though, that's when it all clicked. I applied to analytics consulting firms and the rest followed.

Is this when you went to Booz Allen? What did you do there?

Yes. I went to Booz Allen and did a lot of analytics consulting work. I was on a team called Modeling, Simulation, Wargaming, and Analysis which exposed me to a huge variety of analytics approaches, techniques and problems. One month I'd be doing system dynamics modeling, the next month I'd be building an optimization modeling tool whose GUI was a bunch of Gantt charts. You never knew where the next project would lead.

From there, I went on to do consulting at a boutique consulting firm called Revenue Analytics that does pricing models that adjust prices on hotel rooms, cruises, etc. These models are complex IT projects, so most of the clients who had the data to power them and could afford them were Fortune 500s.

During this stint, I worked with Coca Cola in Shanghai to build an optimization model that pulls frozen barrels of orange juice pulp from oranges sourced all around the world and blends them together so that every time you drink one of Coca Cola's Pulpy drinks in China, the feel of pulp in your mouth is consistent. The project felt like discovering some bizarro corner of the analytics universe halfway around the world.

All these Fortune 500 projects were really fast-paced. But from there I jumped to MailChimp, which is more of a startup, and nothing in the Fortune 500 world could have prepared me for MailChimp's pace. We're on a release cycle where every four weeks, we're putting out a new version of the application. That's light speed for me and, in fact, it's too fast for a lot of data science projects, especially if you have a lot of infrastructure requirements. I'm the slowpoke of the organization. That's an exciting place to be because it means people are pushing me.

One fascinating aspect of MailChimp as a startup is that it's based in Georgia. Not in Silicon Valley or even New York or Boston. What is the startup scene in Atlanta like?

The startup scene is alright because Georgia Tech produces a lot of talent in the Atlanta area. Some of those folks want to stick around our fair city. But that isn't to say there isn't a massive magnet out at the West Coast, because people want to go out to the Valley, join a startup, get equity, and see if they can cash in that lottery ticket later.

That's a very different culture than what you find in Atlanta.

That's something that we have to think about when we recruit, so we play to our strengths. We have some of the most amazing data sets in the world. Two of our domains are in the Alexa 500. We send ten billion emails a month and process another three billion events on top of that. We added 200,000 active sending customers this quarter. We're growing so fast, and the nice thing about that message is it attracts those applicants who want interesting work rather than those who merely want an opportunity to cash out later.

How does the company think about staying in Atlanta?

There are a couple of advantages in being where we are. What I found is that if you're in

the Silicon Valley, you can be part of a conversation that's occurring between all these companies, and there are advantages to that because you know where things are headed. There's also a disadvantage, because you lose a lot of mental freedom.

In fact, it can instill a lot of fear.

What I found is that if you're in the Silicon Valley, you can be part of a conversation that's occurring between all these companies, and there are advantages to that because you know where things are headed. There's also a disadvantage, because you lose a lot of mental freedom.

You hear a lot of what other people are doing, and it's like being on Facebook where everyone's projecting the best version of themselves. This puffery makes you depressed, and you flail about to technologically keep up with the Jones. MailChimp doesn't have that perspective, because we are slightly isolated. This isolation allows us a little breathing room to seriously evaluate technologies, opportunities, markets, trends, etc., rather than just jumping head first into something because everyone else is doing it.

That said, the folks at MailChimp get around *a lot*. I travel nonstop. I speak a lot. I meet with companies. I have conversations constantly with folks around the world, but it's targeted and intentional rather than getting an earful all over the place because you live in Silicon Valley. What that means is that there's less fear, so we're not thinking, "We have to take VC money" or "We have to acquire this start-up."

Talking more about unconventional thinking, you've written in the past that "Your model is not the goal; your job is not a Kaggle competition." Can you talk about why you don't think Kaggle is where data scientists should be spending their time?

There's nothing wrong with Kaggle. I think it's a great idea. If a company's at that point where they want a model that's that good and they're getting a lot of revenue and want to push like Netflix, go for it.

My one criticism is that the way journalists write about it gives a skewed view of what data science is. There was an article on GigaOM where the author said, and I'm paraphrasing, "The main thing data scientists do is build predictive models. That's how they spend most of their time." This is a myth that something like Kaggle will perpetuate.

Before you build a model, you need to know what data sources are available to you within the company, what techniques are available to you, what technologies are available, you have to define the problem appropriately and engineer the features. Usually, when you

grab data from Kaggle, all of these steps are done for you. You don't have to go around looking for data. You can't say something like, "Maybe they left some data behind. Can I come into your company and look around?"

I feel that there's so many steps before you get to modeling that are crucial. Can I ever ask a Kaggle competition, "Is this the competition this company should actually be having?"

Think about the Netflix prize. They were trying to predict what star rating readers would give a movie given past data, but I think they backed off that a little bit because they noticed it's not all about five-star movies. For example, I watch garbage. I will give it two stars, and I will watch it anyway. It's more about moods. A lot of things drive viewership, such as what my friends are watching on Facebook. That's something Netflix is doing now — and it's made their original modeling endeavor somewhat irrelevant.

There was an article on GigaOM where the author said, and I'm paraphrasing, "The main thing data scientists do is build predictive models. That's how they spend most of their time." This is a myth that something like Kaggle will perpetuate.

So there's this notion in data science about whether or not a project should be tackled in the first place that is *a priori* ignored by Kaggle. And I think a big component of data science is questioning why you're doing what you're doing — choosing problems to solve while rejecting other problems that are irrelevant to the business. With Kaggle, for better or for worse, that job is done for you. Kaggle is just an exercise in using a data scientist as model-building machine.

I still think that Kaggle competitions are awesome, and I will never match the intellectual ability of some of the competitors on that platform. I just like to emphasize the other fundamentals of operating in a data science role at a company. I wish there was more focus on them, but those aren't really sexy to talk about in the media.

What are some of these other fundamentals of operating in a data science role at a company?

Well, one of the fundamentals that everyone talks about is cleaning and prepping data yourself. Finding, pulling, prepping, cleaning, the list goes on. Data manipulation prior to model building is huge. But let's go beyond that.

For me, a core skill that any data scientist should possess is the ability to communicate with the business. It's dangerous to rely on others at a business to actively identify and throw problems at the data scientist while he or she passively waits to receive work.

When that's the setup, the business often hands over the wrong problems, because other teams have no idea what data science can help and what it can't.

But if you've got a data scientist who's good at communicating, then that data scientist can actively engage in conversations with the business and with executives to prioritize how to best use analytics.

I believe a good data scientist is one who's engaged enough in conversation with the business to say, for example, "Hey, I know you guys think social data is cool, and I do too. But only 10% of our customers are on Twitter, and it's anything but a random sample. Have we considered using this other transactional data source to approximate what you want instead?"

So now we've got two skills that are important other than building models: data manipulation and communication. What else?

I think a big component of data science is questioning why you're doing what you're doing — choosing problems to solve while rejecting other problems that are irrelevant to the business.

There's one skill that I like to harp on: the skill of editing. People have a strong desire to distinguish themselves from the herd by flexing their expertise. We see this in all industries and jobs. If you have a particular knowledge set, you're going to show that off. In analytics, the way that tendency manifests is by making models overly complex. And by that, I don't mean "using a complex model when a simple one gives the same performance."

No, I mean "using a complex model that is brittle and overly burdensome for the organization to maintain, i.e. whose likelihood of abandonment is high, when a simpler model has a better chance of long-term survival." Sometimes that means using a simpler model even when some performance is lost. That takes an editing eye. And in data science, as in many disciplines whether that be journalism, oil painting, or speechwriting, editing distinguishes the experienced practitioner from the newbie.

One of the big ideas you mentioned is the fact that complex model-building is not what a data scientist spends most of his time on.

Do you think, in the future when there are more tools built for data scientists to take care of all the steps before modeling, data scientists will in fact be spending most of their time on complex modeling?

I think we're already seeing the commoditization of a lot of these skills. It's not that hard to read a book on R and learn how to build models. It's pretty easy, and that's where online education can come in and fill in a lot of technical gaps. If that's all you need as a business, I have faith that not only can cheap labor fill in that gap, but tools are eventually going to get there, too.

The part that will be irreplaceable is knowing what's possible from an analyst's perspective. So, there are a lot of unsupervised techniques, but knowing these techniques and identifying data and opportunities within the company where those opportunities can be married with the data is not purely a technical problem. It's a creative problem. It's knowing all these things and being able to connect the dots. I feel like that's going to be a very human problem for a very long time.

This is related to what you've said before — that a data scientist is a Renaissance figure because it's connected to sociology, ecology, business, computer science and math where you put it all together to solve problems.

It's dangerous to rely on others at a business to actively identify and throw problems at the data scientist while he or she passively waits to receive work.

Right. One of the things I've noticed whenever we try to hire data scientists is that the most effective data scientists are the ones who can communicate effectively. They can talk to people. They can communicate in writing. They can craft an e-mail. They can craft a document that's technical while also lucidly explaining what they're doing. They can tell a story. Those are skills that are refined by studying, wrestling with and arguing ideas across disciplines. And when I encounter data scientists who often enter the discipline tangentially through a variety of other disciplines, I see this breadth and the way it facilitates communication.

Now, why do I think that's important?

Just as crucial as data cleaning is to the beginning of a modeling engagement, communication in the form of change management is crucial during and at the end of a modeling engagement.

Change management is the idea that after you build a model, how do you get other people to use it? You don't just walk into a business and say, "I built you a model. Trust it." There are issues around working with people, communicating, and understanding their context that don't come from just learning to do data modeling. That's a completely different skill set and it's one that the Renaissance person (i.e. an employee with a wide breadth of study and experience) is more likely to possess. If I'm going to be telling

stories and communicating to a wide group of people, I need more training than just being siloed in the math department.

So I try to hire people who look like that. They can do all these things. They can talk to people. They can write spaghetti code. They can push around data. They can build prototypes.

We're seeing this Renaissance-person idea, i.e. this multidisciplinary, quasi-quantitative, quasi-liberal arts approach affect a whole host of disciplines. Just look at the new "digital humanities" movement where things like the Trans-Atlantic Slave Trade Database are being built that historians, linguists, etc. are setting their SQL on.

Just as crucial as data cleaning is to the beginning of a modeling engagement, communication in the form of change management is crucial during and at the end of a modeling engagement.

I love that we're seeing data and the humanities collide, and that's happening in business, too. That's why we have this weird hybrid concept of a data scientist who's not really a scientist. When you look at the preeminent data scientists out there, they are not people who are just in a lab acting out your canonical, stereotypical view of a scientist. A lot of them are writers and speakers and executives and a whole host of other things than your typical white-coated scientist.

At the same time, there exist a lot of the people who are doing that are in a Ph.D. or are considering a Ph.D., and one of the reasons they're attracted to data science is because they've heard that it applies to their researching skills in the industry. Given all the things you mentioned that couldn't be acquired by staying in the math library, do you think this idea of the data scientist as an applied researcher is a misconception?

There are certain companies where there's a resemblance between the two. For example, if I'm going to be doing ad targeting on Facebook, I'm going to be part of Yann LeCun's new deep learning lab. I imagine for that type of data science, academics are going to find that a fine transition.

However, there is a vast array of companies now that think they need data science talents, and the data science talent they need is not someone who has been specializing in one particular academic area for six years of graduate school.

That's not what companies need. They need someone with a broader skill set.

I've seen too many Ph.D.s go into companies with a not-my-job mindset where they're going to wait for you to bring them a problem that fits perfectly with their expertise. If you don't bring them that problem, it's not their job. I get it — they fought hard for that doctorate.

But this is a dangerous way of thinking that could sour a lot of people in the industry.

I like people who are more aggressive and want to find problems to solve. Maybe they don't get to use the techniques they've used in the past, but they know there are a lot of analogous concepts in these techniques. For example, one of the things I put out in my book is that machine learning algorithms, whether they're unsupervised data mining techniques, AI modeling, or forecasting, all have an optimization component.

The point I'm trying to make is that even though you are focused on one thing, all these things are related. All these concepts are related. Cluster detection and outlier detection are two sides of the same coin in a graph, and I try to tie them together to show people that if you can do one of these things, you can do all of them. You should be excited to learn all these things and figure out which ones you can use forever.

You're like a kid in a candy store where you've got all these opportunities to do these things. Those are people I would love to see move into this industry. Some of those folks are Ph.D.s, but sometimes the specialization that comes with too much time in graduate school can be a burden.

As more people move into and understand data science, do you think that the future will bring data, and statistics, literacy for the masses?

Knowing how slow academia moves, it's going to take some time to get there. I went to the University of Georgia, and everyone there had to take a math class where the textbook had a cover with Waffle House on it, which shows you the level of math they were learning. I think we're moving into a world where people need to know more math, and it's no longer acceptable to say, "I'm not good at math. Math isn't for me."

Everyone's going to have to be literate. When I worked in management consulting, I met a lot of strategy consultants who came from non-quantitative backgrounds, but every single one of them knew how to do a pivot table. They knew how to write a VBA macro and filter data. They knew the basics of how to move data around in a spreadsheet. They would never call it math or programming, but it's pseudo-math-programming. Oddly enough though, those simple skills were an essential part of what the client was paying for.

I think there's going to be more of that need in the future. People are going to need to know how to do things like significance tests, sample size calculations and so on. We need to find a way to fit this data literacy into a liberal arts curriculum. That requires motivating the concepts.

People are going to need to know how to do things like significance tests, sample size calculations and so on. We need to find a way to fit this data literacy into a liberal arts curriculum.

In "*Data Smart*" I try to motivate people to learn these techniques as much as possible by showing how to explicitly use them in business. The cool thing with teaching people later on in life, once they have a job, is that the motivation totally clicks. When it comes to teaching students, especially those majoring in something else, they're unmotivated. School's never really cracked that nut, but I think it's headed that way. People need to be data-literate. There's no way we're going to get by without it.

At the same time, there's a debate in some parts of the Valley. There are people saying that numbers are pushing out the usage of human intuition, and that there's an over-usage of analytics, where you're AB testing every shade of green on your button and you go with whichever one performs the best. As the future becomes more driven, what's your take on this type of criticism?

I do agree, and I think that approach is dangerous. At MailChimp, we often make fun of Key Performance Indicators, aka KPIs, which are the lifeblood of the corporate analytics world. When we listen in to the quarterly calls our competitors have, we can see they're driven by metrics like ARPU (average revenue per user) so much so that they've lost sight of things that are not unimportant but are just harder to measure.

You're optimizing average revenue per user not because it's the most important things but because you can measure it and Wall Street can measure it and look at it. That's a way to grade your company, but what does average revenue per user mean when there are users on your Facebook site saying, "I fucking hate you guys"? That could be a red light that something is wrong, but you're not paying attention to it because it's not a metric you care about.

I think we should leave room for people to be creative and to think about soft things like customer happiness. At MailChimp, we purposely don't measure a lot of metrics against our marketing team. Our marketing team has a budget, but we don't look at things like conversion. We took out billboards in cities across the US, and the billboards just have a picture of Freddie, our chimp mascot, with a blue background and no words at all. The only people who really understand what it is are already MailChimp customers and maybe our competitors.

We can't look at conversion of that, or how it affects revenue. That's not something we're interested in. It's about giving our users a good experience. They see a billboard on their way to work, and they're thinking, "*Aww... That's MailChimp.*" And there's value there. It's like an inside joke in a subtle way. I might go to a conference and have a MailChimp user come up to me who's excited to meet someone from MailChimp. They might say, "I love using your site. It's so much fun. It's one of the best sites I use for my job." That's great. That's a kind of person who's going to go tell other people about our product, but we don't have to measure it.

At MailChimp, we often make fun of Key Performance Indicators, aka KPIs, which are the lifeblood of the corporate analytics world.

Its better that we just keep delighting customers so they tell other people about the product rather than AB-test button colors. I'm perfectly happy to leave things in the hands of talented designers, people who are not quantitative but know what they're doing.

Have you ever heard of Tony Hsieh, Zappos CEO, Downtown Project in Vegas? He moved the headquarters of the online retailer from the Silicon Valley to Vegas. His perspective is that it's important to engender serendipity but not with contrived methods. Although he runs a tech company, he's much more open to the intangible things such as human creativity and experimentation.

It's interesting having two well-known companies in very different parts of the world that are technology-driven but not in Silicon Valley, and as a result they think of things differently than other firms. Do you think there's any relationship between being not in the Valley and being able to think the manner you described?

Honestly, part of that for us is that we are privately owned. We are not seeking to go public, and we're not taking funding from any other companies. We have the freedom to be creative, because there's no one breathing down our neck.

MailChimp's bootstrapped, and, because of that, we have immense freedom. We're not trying to sell the company to someone else. When your goal is to sell your company, things can get perverse. You get distracted, and that is dangerous from a competitive standpoint. If we get distracted, we might check out or lose sight of what other competitors are doing. Part of our different perspective is driven by being outside of the Valley. But another large part of it has been knowing that we're not taking funding.

A lot of people these days are interested in *starting* companies — being founders, etc. We at MailChimp are interested in being a company in the long-term. That looks very

different, and I'd argue it's a better place to be as a data scientist. When you're a data scientist at a young company looking to go public or be acquired, then your work ends up getting commandeered for marketing purposes. It becomes difficult to invest in analytics that might have long-standing customer value versus some short-term *wow!* factor.

Wow. That's a pretty amazing distinction between MailChimp and other tech startups. I've heard of one project at MailChimp called the Email Genome Project. Can you talk more about that?

The Email Genome Project was essentially an infrastructure initiative at MailChimp to create a dossier for every e-mail address we've ever seen and store data about it. In fact, right now it resides in RAM. It's one of the largest in-RAM databases in the world. We use Redis to do it, so it's essentially a big Redis key-value store summarizing interactions we've had with about three billion unique e-mail addresses.

MailChimp's bootstrapped, and, because of that, we have immense freedom. We're not trying to sell the company to someone else.

We built APIs around this data store and use those internal APIs to power data products. We have an anti-abuse AI model called Omnivore, and that runs off EGP. One of my favorite internal products is called NotABot. When users sign up for MailChimp, we check NotABot, and if you look legit, we hide CAPTCHA because of everything we know about you. We say, "We've already looked at your behavior. We know you're a human, so you're good to go and we're just going to hide the CAPTCHA."

The funny thing is that the data science project is not something built in D3. It's not something cool with bubbles. Literally, this data science product is *the absence of something*. All I've done is taken CAPTCHA away, and I feel very proud of that. Removing things improves the user experience; this is one way to make users' lives suck just a little bit less.

We had CAPTCHA in front of our help form to contact support, but when you want to contact customer support, you already have a problem. It's a perfect opportunity to reduce friction for these confused people rather than adding some shitty CAPTCHA icing to their confusion cake. That's a small project we've done, and we've done a lot of things like that using EGP's internal APIs. We tell people, "Here's what this API call does. Here's the data that backs it." Then we expose the API call to the devs and see what happens.

We did another project called Send Time Optimization where we noticed a couple of things. One was people asking what time they should send. People were going online

and just reading anecdotes. It's not like all your customers go to work from 9 to 5 and take lunch at noon. But those are the kind of assumptions you'll see in anecdotes from supposed marketing gurus.

One of the things data science promises is that we can provide people's personal experiences. Using EGP, MailChimp can tell you about *your particular subscribers*. What do you know about their behavior? If you're writing to line cooks who work the night shift, they're probably not awake at 2 PM. So what Send Time Optimization (STO) does is it pulls all the records for these email addresses (even if we have new e-mail addresses, we've seen them before due to other MailChimp email they've gotten), and using those records, STO hands you a send-time recommendation. Anecdotes are for chumps.

How does MailChimp then use data science to power these personalized product features?

So far I've laid out a few of MailChimp's data science products: Omnivore for anti-abuse, Send Time Optimization, and NotABot. But we've got a lot more. For instance, we use AI models trained on past interactions with customer support to make knowledge-based article recommendations. We use data mining algorithms to find segments on people's lists and suggest those segments to them. We use optimization modeling to schedule our customer support employees to meet forecasted ticket demand. We use a lot of Holt-Winters with prediction intervals when making infrastructure forecasts.

Some of these products are supervised machine learning products, others are classic ops research products, graph mining products, forecasts, etc. We use whatever techniques and whatever data gets the job done.

Some products are user-facing, some are internal. Some are big and require tons of infrastructure. Others, like our likelihood-to-pay model, are nothing more than a logistic regression whose coefficients fit in a single short vector.

So how do we use data science to power these products? Any way we can.

The one common theme these products have is not an approach or a data source or a technology. The one common theme is that each product solves a problem for the business or the customer. I run my data science team like an internal consultancy. We're all about being useful.

There's this joke going around making fun of data scientists that says that a data scientist is just a statistician who lives in California who calls himself that to get a job. Given that you're someone who is both professionally a data scientist, yet at the

same time seems to share a sense of skepticism about things that are overhyped, what is your take on the burgeoning field of data science?

I think the term "data science" is somewhat ludicrous. The phrase "Data science" is two vague words glommed together that don't actually describe most of how I spend my time. Data science as a term may die, nothing but a fad title, but the skill set is so important that it will spread into many roles within the business. It wouldn't surprise me if, a few years from now, most MBAs require a couple of data science-style classes.

The field is just going to get into the water.

The more you investigate data science as a category, the more you see it's an umbrella term disguising insane amounts of variety in skill sets and backgrounds.

The more you investigate data science as a category, the more you see it's an umbrella term disguising insane amounts of variety in skill sets and backgrounds. A data scientist is unlike a stonemason in that way. There isn't one background for data scientists and there isn't one thing that we do. We've seen data scientists who are more data engineers. We've seen data scientists who are AI professionals. We've seen data scientists who are good at visualization and doing front end development. There are data scientists like me who are nothing more than embedded strategy consultants who like math.

The term could die or fracture into multiple titles, but need for those skill sets won't. Students tend to worry about that and say, "People won't need data scientists by the time I graduate." They'll need *something like a data scientist for sure*, so just call yourself that. My past job titles used to include words like "analytics" and "business intelligence." That's fine. The terms come in and out of style, but if you are good at understanding problems and communicating with people, and answering their questions with data, the need for you in particular will never go away. You will never be automated. You will have plenty of job security.

During a conversation we had with D.J. Patil, he told a story of how, at some point, he consulted for the US government when they were in Afghanistan. He mentioned how there was a lot of chaos and everything was going crazy, but there was a lot of opportunity that came out of that chaos and you could influence people because no one knew what was going on.

Data science seems to be evolving in the same way: where there's chaos, there's uncertainty and as a result, plenty of opportunity. Do you think there are going to be

large opportunities in the future as data and technology become more prevalent?

Yes. I touched on this in an article I wrote about Disney. The "meat space" world, i.e. the one not confined to a screen, is full of chaos and uncertainty, and so there's huge opportunity to take the analytics we've been doing for web companies and move it out of that orderly sandbox and into the physical world.

Obviously, wearables are an immediate example of how that's happening. But humans are being "cookied" in meat space by more than just wearables. Think about Nest (and how much Google paid for it). We're doing all sorts of physical tracking, such as MAC address tracking in stores and appending demographic data to surveillance video feeds, so we understand a bit about your demographics and what racks you go to in department stores.

The physical world is messy and chaotic, nonetheless we can understand people's actions as they move throughout that space. That's where I see the most opportunity for data science.

Disney saw this opportunity when they introduced a long-range tracking component to their wristbands. They track you in physical space so that they can provide personal experiences in the physical world and not online. My kids rode "Pirates of the Caribbean" eight times at Disney World. Then we visited this animatronic Mickey and all he would talk about were pirates, because he knew that's what my kids were into based on their transactions in the physical world.

The physical world is messy and chaotic, nonetheless we can understand people's actions as they move throughout that space. That's where I see the most opportunity for data science.

It sounds like the overarching theme is that the personalization of the internet, of visual space, is going to move towards the personalization of the physical world?

It's going to merge. In fact, the internet is simpler than the real world because I can "cookie" you on the internet. We're going to learn how to cookie people all over the physical world too, and I think people are freaked out about this from a privacy perspective. I agree and sympathize. There's a creepy side to the word "personalization."

It's a frightening affront to our personal freedom. While I've gotta live my work-a-day life, there will be companies tracking me that will be dedicated to getting me to do one thing, like opening a credit card or drinking a Coke. It'll be data-driven asymmetric warfare. They have my data. They know my issues — financial, personal, etc. Their models will

know how to target me. They can pull my strings.

I think that is concerning, but, at the same time, we're doing this to ourselves and share the blame. I install whatever mobile app I want to and just blaze through the permissions. People always say "but it's my data." If you give it up in exchange for a free game, it may not be anymore. The undervaluation of personal data by consumers is endemic today.

So yes, personalization on the internet will morph into personalization everywhere. But we've got to figure out all this creepy stuff as we head in that direction. Part of that will be cultural, and part of that I'd imagine will be legal and legislative.

A term I've heard before is "data superhero", and it's the idea of putting yourself, as a data scientist, into a position where no one in Congress knows what data science is. They've never read "*Data Smart*", and the superhero is the one who knows what it is and is able to inform people and stand up for the public interest.

Data scientists have a particular set of skills and knowledge that makes them essential to business today. A lot of that knowledge and skill is being used to blaze new trails for how we as individuals, consumers, citizens, etc. interact with businesses, our government, our peers. There is abuse and confusion as well as opportunity to fundamentally change entire industries and practices for the better.

Given that, data scientists can take on public-facing roles as subject matter experts. People want to know what's possible with data, both to understand if abuse is possible as well as to understand if progress is possible. Too many people think of data science as magic, but data scientists can come in and bring the discussion back down to earth. We can say, "no that's not possible," and "yes that's possible," and "yes that other thing is possible but you'll need express legal consent from consumers," and so on.

And that's a role we should take up, because if data scientists don't engage the conversation then we should expect voices with less training to fill that information vacuum.

JOSH WILLS Director of Data Science at Cloudera

Mathematics, Ego Death and Becoming a Better Programmer

Fascinated with the beauty of calculus at an early age, Josh Wills majored in pure math at Duke. His first introduction to statistics was in the final year of university, where despite some misgivings of it being not nearly as interesting as hyperbolic partial differential equations, he actually fell in love with the discipline.

After a brief stint at IBM, he returned to do a PhD in Operations Research at UT Austin, trying to solve NP-hard problems. Afterwards, he joined the startup world, working as a statistician first at Zilliant, then Indeed and finally Google.

In this interview, Josh offers beautiful thoughts on the intersection of literature and data science, learning through humility and masochism, profound moments in open source projects, and the deep impact that Google's engineering had on him. Josh Wills is currently the Senior Director of Data Science at Cloudera, where, according to him, he "makes data into awesome."

To start the interview, we'd love to revisit your undergraduate days, from when you were just going into college, to graduate work, and how your experiences led you to where you are now.

I was a math major in college. The funny thing is that I never really liked math when I was growing up, even though I was pretty good at it. I was more into history and political science, until I got to high school and discovered calculus. I was enthralled. I loved calculus and felt that it was the first interesting piece of math that I had ever encountered.

I was a pretty big nerd in high school, which won't be shocking to anyone. I did things like study for AP exams for classes I wasn't actually taking. During my junior year, I took exams for AP Political Science and Comparative Government without actually taking the classes. I did well on those exams, so I ended up doing the same for Art History, Economics, and Physics during my senior year. I also read all of Calculus AB and BC in a semester, and then I got into multivariate calculus and proceeded to linear algebra, all on my own. I was just completely enthralled with the beauty of mathematics, the same way a person would appreciate a beautiful painting or work of art.

I ended up going to Duke University. The best part about Duke was that I got to take whatever math courses I wanted right away. My first course was graduate level topology. That was interesting because I was taking it with the other math freshman who were really good mathematicians. It became apparent to me relatively quickly that I while was good, the other freshmen were on another level altogether, which was very humbling. I think everyone runs into this at some point in life, and I felt relatively lucky that I encountered it during my freshman year because it gave me time to recover.

I was just completely enthralled with the beauty of mathematics, the same way a person would appreciate a beautiful painting or work of art.

Anyway, I stuck with math, and I thought I was going to become a math professor. But I was also interested in many little side things- I did philosophy, economics for a while, and then became interested in cognitive neuroscience. I was lucky enough to do a Research Experience for Undergraduates (REU) fellowship at Carnegie Mellon the summer after my sophomore year, modeling road and spatial navigation. That was my first introduction to real programming in MATLAB, building large models to simulate brain function. That experience is what got me interested in programming in general.

Did this push you to start taking programming classes at Duke as well?

Yes. I took Duke's introductory courses in computer science and I learned how to program in C++. I never really studied algorithms or operating systems or other things computer science majors study. In my professional career, I've discovered all of these huge and embarrassing gaps in my computer science knowledge, usually during job interviews.

At the start of my senior year, I decided to put the academic career on hold and go get a real job. I was interviewing with some startups and accepted an offer, but it was rescinded as part of the whole dotcom implosion thing that was happening in late 2000/early 2001. I wasn't alone here, and Duke's recruiting office was really great in helping folks find jobs elsewhere. I ended up getting a job in IBM's Austin office. My first day was June 17th, 2001, and the week after I started, IBM announced a hiring freeze, so I suppose I slid in just under the wire.

IBM Austin had a hardware group that does chip design and system bring-up, which is where you hack early stage hardware to get around all of the bugs so that you can load and run an operating system. I was managing a MySQL database of test data for microprocessors. All in all, it was 15 gigabytes of data, which at the time seemed enormous, but now seems laughable — my phone has more storage than that whole volume of test data! I was building dashboards and running statistical analyses of machine

performance and chip performance; trying to predict how fast a chip would clock based on a number of measurements that were made during wafer fabrication. It was classic statistics, classic data analysis, and just learning how to program. To be honest, it was pretty dull, and I got bored with it fairly quickly. I also have this masochistic approach to achievement, and so sometimes I like to do things just to prove that I can do them, regardless of whether or not it's actually a good idea or not. So in that vein, I applied for and got into the Operations Research (OR) graduate program at the University of Texas at Austin (UT). UT didn't have a statistics department, which is what I actually wanted to study, and OR was as close as I could get without having to leave Austin, which was just a really great place to be at the time.

As an undergraduate, I didn't take any statistics courses at all until my very last semester, which was really my blow-off semester. It was when I took music appreciation, introduction to logic (oddly enough, a philosophy course), and introduction to statistics. Intro stats was actually a requirement to graduate, but I felt like it was beneath me after all of the abstract algebra and hyperbolic partial differential equations. And the funny thing is that I completely fell in love with it. A lot of the philosophy and neuroscience stuff I was into were things involving epistemology and symbolic reasoning, about understanding how we can say that we know something to be true.

And statistics is about quantifying uncertainty and what we can't know.

Precisely! It is the quantification of what is knowable and what is not. Here is your data, what can you say that you know? It was deeply appealing to me. Personally, that kind of stuff really winds my clock. I loved statistics. Fast-forward a couple of years, and now I'm at UT and taking a full graduate course load in OR. I did three courses a semester for two years to get my Master's degree, while simultaneously working at IBM. That was a terrible idea. It was absolutely horrible. I had no life.

It sounds like you learned how to do the relatively simple statistical analysis at IBM and thought, "I want to expand my intellectual horizon."

Very much so. My IBM introductory software engineering job was pretty easy, and I wrote a bunch of crazy Perl scripts that more-or-less automated my job. But I had this kind of residual itch from my statistics class and from seeing that statistics was actually pretty useful to people in the real world. My mental model at the time was that if you wanted to learn more about something, school was a pretty good place to do it, and so I went back to school.

A semester into my graduate program, I made another switch: I changed teams at IBM to be able to do some "real" programming, not just dashboards and Perl scripts. I switched

to a team did very low level firmware programming in C++. This was basically writing firmware for hardware systems that didn't fully work yet because they haven't debugged all the circuits. I was working as part of a team and learning to use things like source control, write tests, all of those good practices that I never learned in school. More than anything though, the most useful skill I learned was how to debug black box systems. I was trying to run firmware on a piece of hardware that didn't really work yet, and my job was to figure out a way to make that software run by hacking around whatever bugs I came across in the hardware itself. I didn't know anything about hardware. I still don't know anything about hardware. I can't even program a VCR. I think that I became a software engineer because I can't understand any system that I didn't design myself.

Anyway, the black box system is a piece of hardware that doesn't work. I would give it an input, and it would not give me an output. I had to figure out a hack, some sequence of commands, that would cause this piece of hardware to begin communicating with the rest of the system. And this skill, the art of debugging something that you don't understand at all, is maybe the most useful thing I learned there.

What did you end up learning through this experience of debugging black box systems?

It is the quantification of what is knowable and what is not. Here is your data, what can you say that you know? It was deeply appealing to me.

I don't think there's any secret to it: I'm obsessive. I was one of those kids that played with Legos for five or six hours straight. I'm still pretty much like that. I was born in 1979, so I'm borderline millennial. It is unacceptable to me for a computer system to not do what I want it to do. I was willing to beat on the black box hardware for whatever amount of time was required to make it do what I wanted.

I've had a few instances in my life where I have worked on a very satisfying problem. A satisfying problem is one where your technical skills are good, but the problem is just a little bit too hard for you. You're trying to do something slightly more difficult than what you already know how to do, and that is great, great feeling. I can lose myself in those kinds of problems. That's typically when my personal relationships tend to fall apart, because I'm not really paying attention to anything else.

There was this trend for awhile in data science job interviews to have candidates analyze real datasets during the interview. I'm a huge fan of this practice. I had one job interview where they gave me a problem and a dataset and two whole hours of quiet time to just sit and do data analysis. It was maybe the happiest two hours of my entire year. I should do more job interviews just so I can do that.

You had mentioned how, at one point of your college career, you were burned out from academia. One of the hallmarks of academia seems to be that once you've reached a certain point, you have the opportunity to spend all of your time obsessing over an open problem. Given that your personality type seems to fit that role, why wasn't academia appealing to you anymore?

As a pseudo-millennial, I'm not just entitled, I'm impatient. I don't think the requirements of academia were appealing anymore; there were large sets of things I would have to complete before I reached that point of being able to obsess over an open problem.

Once you're a graduate student, you work for a professor on that professor's grant, doing largely what the grant says you're supposed to do. Then, you do a post-doctorate for a couple of years and become an assistant professor. You go through that horror, and, after 10 years, you get tenure. It's a really long time to wait before you can have that promise of obsessive problem solving fulfilled. Even then, I don't feel the promise is fulfilled because you have to spend a lot of time working on grant proposals and managing your graduate students and postdocs.

Now I'm 35 years old. Time-wise, I may be roughly at that point in my career now. I have a really great job where I get to do what I want and do, whatever is interesting to me. But it's also a be-careful-what-you-wish-for situation. The freedom to work on whatever you think is interesting is stressful because there's no one else you can blame if you're not working on the right thing or if you miss a technology shift that has a profound impact.

Amr Awadallah (Cloudera's CTO) wrote a blog post about what a chief technology officer does. He was comparing the CTO's performance to CFO's performance. The CFO is not responsible for making the sales numbers every quarter, but if there is a big surprise miss, the CFO gets fired. Similarly, the CTO is not responsible for shipping products on time, that's what the VP of Engineering is for. But if the CTO misses a major technology shift, he or she gets fired.

I have a CTO-kind of job right now. I am free in my job to think about analytics, the future of data science, what exactly is coming down the pike. If I miss something, I should be fired because that miss could have profoundly negative consequences for Cloudera.

There's tremendous pressure that comes with that freedom. Now that I get that, it's slightly horrifying. I have a fair amount of anxiety about it.

Can you talk a little bit more about what happened in between IBM and Cloudera? How did you get to this point?

We skipped the part of graduate school when I was taking a class in price optimization.

One of my professors worked with a local startup in Austin called Zilliant. I wanted a job focused on operations research, so my professor hired me to work as a data analyst there. There, I went back to SAS and R and started doing data analysis and building models for things like market segmentation and price elasticity.

When you come from academia, you tend to think the world is more interesting than it actually is, or that a problem is more complex than it is. The reason that price optimization hasn't really taken off as a software discipline is because the primary pricing problem for Fortune 500 companies is to sell things for more money than it costs to make them. If they don't know how much it costs to make things, they can't know how much they should sell those things for to ensure that they make a profit. It's not rocket science. You don't need a data scientist to do that. You just need good reporting.

When you come from academia, you tend to think the world is more interesting than it actually is, or that a problem is more complex than it is.

Why is it that companies don't know this bit of crucial information?

It seems like a fundamental component, and yet many of them do not actually know. The problem is incentives. The person who is selling the deal, the salesman, is going to get a commission, and his or her income depends on the commission. They're putting together a package of things that are going to be sold as a part of the deal. There's going to be some materials and professional services, that's just text and contracts. These contracts get read and improved, but no one necessarily understands how much it's going to cost to fulfill these contracts. There's way too much variance. And people have a tendency to be very optimistic. They don't think they're going to have conflicts. They don't think they're going to have errors. They don't think there are going to be hurricanes.

These aren't trivial problems, but they're also not the kind of problems that are amenable to the complicated data analysis techniques that you typically learn in graduate school. They're very different kinds of problems.

They are simple problems. They're simple but not easy. Losing weight is simple but not easy. Most industrial problems are simple but not easy.

So after Zilliant, did you make it your goal to attack the industry problems?

I like to be useful more than anything. I like to solve people's problems. I like to be helpful. I'm a helpful person by nature. I enjoy abstractions. I enjoy art and weird stuff aesthetically, but I would rather have my day-to-day work be more focused on people's problems and making their lives better. The beauty and the theory are never so appealing

that they manage to draw me away from the real problems.

You worked at a bunch of different startups before Google. Were you solving different initial problems at these startups? What prompted the shift to Google?

It really took me forever to leave Austin. I could make a list of all of the bad financial decisions that I made because I was too afraid to leave Austin. I had a job offer from Google to be an engineering analyst, which I turned down in 2005. I turned down a data science job at Facebook in 2007. I try not to think about that one too much.

The thing that finally got me to San Francisco was auction theory. I was working on my PhD at UT and had taken some classes in game theory and mechanism design, and we covered auction theory. I absolutely loved it; it was beautiful math that could also be used to create socially optimal outcomes. I was really curious about how auctions worked in the real world, but there weren't really any places in Austin where I could go design auctions for a living. I was fortunate that I had kept in touch with Diane Tang, who had tried to hire me at Google back in 2005 and was running Google's ads quality team which was responsible for the ad auction. She's now Google's first and only female Google Fellow, but at the time, she was just my friend who hired me to go to Google and work on auctions full time. She has been an amazing mentor to me, one of the most important people in my career.

They are simple problems. They're simple but not easy. Losing weight is simple but not easy. Most industrial problems are simple but not easy.

What was it like on Google's ad quality team? Was that a confluence of smart people who had studied auction theory as well and then implemented it in the real world?

I think the thing to know about Google is that it is smart software engineers with no specific expertise designed most of the core systems. Eric Veach, who had a PhD in computer graphics but no machine learning experience, designed Google's original machine learning system. Eric was tasked with the problem, read a book, and came up with a wholly new solution.

I remember when I first got to Google and read about how that system worked. It was the most brilliant and unique solution to the world's first truly large-scale machine learning problem. His original algorithm was really clever and I've never seen anything like it published anywhere, and I don't think we ever will because, of course, Google has now gone on to even more advanced machine learning systems.

Eric was also the person who designed Google's original auction algorithm. Again, Eric

is a graphics guy, he's not an auction theorist. So he read a book about second-price auctions, and he came up with this very simple generalization that is called the GSP, the generalized second price auction.

I worked on a number of auction-related features and launches at Google. I really enjoyed it, but at the end of the day, the auction can only be as complicated as the understanding of the auction participants. Advertisers are wonderful, but they're still just people, while the really interesting bidding strategies and auction models are so complicated and computationally intensive that they require serious software engineering chops just to participate in them. It wasn't in Google's interest to have an auction that was so complicated that no one besides auction theorists could appreciate it.

This seems to be emblematic of one of the differences between academia and industry. In academia you're focused on getting the optimal solution. In the real world, you find that your implementation priority queue is dominated not only by optimality but also by feasibility and expedience. Was this shift hard for you to see and interact with?

I don't think so. I was fairly lucky. Most of my graduate work in operations research was working on impossible problems. Operations research consists primarily of very hard problems where you cannot find the optimal answer. The job is to do the best you can, and I actually love those kinds of problems because the expectations are low. If the problem is impossible and you are able to do anything even remotely close to a good solution, it's kind of amazing.

There is a joke: "If you have a NP-hard problem and you make it slightly better, your solution is exponentially better"?

I could not agree more. It was a good headspace to be in. Operations Research is a very practically oriented science and academic discipline, so transitioning to that industry mindset was not one of my problems.

Your story illustrates that to be a great data scientist, you have to be slightly masochistic. You have to be willing to go out of your way to be in areas where you're the least skilled person in that domain and programming. What was your development like as a programmer?

I may not be able to give myself too much credit. I was a pretty good programmer in school with algorithms and optimization routines, but I wasn't really a great team programmer. Even at IBM, although I was on a team of four developers, we really didn't have to work all that closely together. The structure of the software was already well-defined and the interfaces were clear.

When I was at Zilliant, the company decided to redo their pricing engine. The data analysts got together and wrote a spec about what they wanted the pricing engine to do. It required some domain expertise, and of course I had programmed for many years at IBM. So I was tasked with doing the implementation, but it quickly became clear to basically everyone that I did not know how to build a real software product from scratch.

I give the managers at Zilliant a lot of credit for what they did next: they apprenticed me to a much more senior developer, John Adair, who is another great mentor and friend. For three months, he implemented the spec, and I unit tested it. I wrote unit tests and integration tests for his code every single day for three months.

It was the most useful learning experience of my professional life, because John writes beautiful code. When I describe this experience to people, they always make a face, because it sounds tedious and awful and lots of developers hate writing tests. But when it's your job, and you're measuring yourself the whole day, it can actually be fun. And I was just learning so much about how you actually build systems from scratch.

I was somewhat involved in writing the spec for what the software was going to do, so I knew both the spec and the software very well. What was interesting was was getting to see how to write code that is designed to be testable. John and I went through a few refactorings over the course of the project, but the QA team only found like two bugs when the system was tested. It was the best software I've ever been a part of. It was beautiful code.

When I describe this experience to people, they always make a face, because it sounds tedious and awful and lots of developers hate writing tests. But when it's your job, and you're measuring yourself the whole day, it can actually be fun.

After I left Zilliant, I did a brief stop at Indeed, the job search engine. There, I was a statistician. I wrote some code, but I was primarily there in my capacity as a statistician. And when I left Indeed to go to Google, I was also hired as a statistician. For whatever reason though, when I actually got in the door at Google, all I did was write code. There was just so much great code everywhere that you could read and use and learn from. After nine months at Google, the company changed my job title from statistician to software engineer, and even gave me a promotion. I've always felt a little shady about that, because there's basically no way I could have ever passed a Google software engineering interview.

For someone like me, I am really just a good mimic, and I can pick things up pretty quickly. Being inside Google with so much good code, was an absolutely amazing experience. I am 20 times better as a software engineer just because of my time at Google and seeing

what the people who are really good do. It was an unparalleled experience, absolutely amazing.

Can you give us specific examples of how you did that? Do you go to the people that wrote code, understand the problem from them, see how you would implement it and read it? What was your procedure for learning all that you did?

I don't know how other places do it, but Google imposes it on you. They force you to code the way Google codes. Readability standards are a big deal. You have to get readability in any language that you want to be able to commit to Google's source code repository, or have your code approved by someone who does have readability. To get readability, you have to write a large chunk in a way that adheres to Google's coding style, and the process of readability reviews is basically hazing for software engineers. I will never forget my readability review for Sawzall.

I was writing some code to analyze the ad logs, studying correlations between advertiser bids and various machine learning probabilities that we were calculating. I wrote some basic correlation routines and then submitted them to the core Sawzall libraries, and it turned out that my code reviewer was Rob Pike. If you don't know Rob, he's an old school AT&T Labs guy. He wrote Plan 9, and he's the creator of the Go programming language. He also created Sawzall. He's also the most pedantic code reviewer I had at Google, and I'm sure that he will consider that a compliment. I think I went through 26 code revisions during that review with him, and it was absolutely awful. It was so bad that I really thought hard about quitting. So, so, so many nitpicky comments. I think that was a great thing about Google, they tortured me into becoming a better programmer by forcing me to think hard about all of these little decisions. No pain, no gain.

That seems to be one of the nice things about being a data scientist. It's at the intersection of many fields, so when you're in a particular field, you can humble yourself by not thinking of yourself as a practitioner of that field and say, "What can I learn from this person as they are a practitioner of this field?"

I think that is a big part of your job description as a data scientist. The reality is that these things are never one-way streets. For every software engineer who gave me a scathing code review, another one would come to me later with a data analysis problem, because they knew me as a statistician who spoke their language and could explain things to them.

It's hard to humble yourself, but keep in mind that it almost always comes back around in a positive way. It's good for your career to come in and be seen as an expert in something who knows how to communicate their expertise.

How did you deal with the transition between a large company like Google, where there's so much institutional knowledge to draw upon, and a startup like Cloudera?

There are lots of things I miss at Google. I miss the people. I miss the food. I miss the toys. They had a lot of great stuff at Google. To the extent that we have a product strategy on the data science team at Cloudera, it's to take stuff that we loved at Google and create open source versions of it. That's all there is to it. It's the easiest product management strategy in the world. Know what you like and try to improve upon it.

When I got to Cloudera, it was roughly 85 people. It wasn't a startup, but it was pretty small. I was like, "Hey everybody, I'm the new director of data science. What should I be working on?" No one had any idea, and I didn't have any idea either. It wasn't entirely clear to me what I was hired to do. I had a couple of days of tremendous anxiety about that. I was completely useless. At Google, I had 150 emails a day from people who needed stuff from me. Here, I could hear crickets chirping. It's that anxiety-inducing freedom we talked about earlier.

Just because I'm not the best programmer in the world doesn't mean I can't contribute useful things, and the community that has sprung up around Crunch is something I am incredibly proud to be a part of.

So my job at Cloudera was to figure out what I could do that would be useful. I spent a lot of time talking to customers, and I still do a lot of that. I give them advice about building data science teams or about particular approaches they can take to solving different types of problems.

I also started working on problems that customers talked about and various customer engagements that seemed interesting and useful. I was also new to the Hadoop stack, and so a lot of it was just learning what was out there and how things worked. I remember one project where I was building a model for detecting adverse drug events using an algorithm that was created pre-MapReduce but that was really a perfectly MapReduce-able problem. That was the first useful thing I did, and I know that because Mike Olson, one of our co-founders, presented the results of my analysis as a five-minute quick hit presentation at a conference and we got a lot of nice press and Twitter coverage for it.

A little later, I was working on a problem that required processing lots of seismic imaging data, which is time series data oil and gas companies analyze to try to figure out where oil and natural gas deposits are located under the earth. That was the first time I really missed FlumeJava. It was the perfect tool to solve the problem I was working on, and so I rewrote enough of FlumeJava to be able to solve my problem.

That process brought me back to my black-box debugging days at IBM. When I was at Google, I had used FlumeJava to write data pipelines, so I knew what the APIs looked like, but I didn't really understand how it worked under the covers, only how it worked conceptually. The FlumeJava team had published a paper about the system, and that was tremendously helpful, but there was still this process of sitting down and saying to myself, "okay, I know the API worked like this. I don't know why it worked like that, so let's see if we can sit down and figure out what had to be going on so that this thing will work."

It really took three times to create the FlumeJava clone that eventually became Crunch. The first time I wrote it, I really coded myself into a corner; I made some design mistakes that I just couldn't unravel. So I started over, but I ended up creating this ridiculously over-engineered monstrosity, and I wasn't really getting closer to being able to solve my original seismic data analysis problem. So by the time I started over again, I really needed to get the thing to work quickly, and fortunately I had learned enough from the first two attempts to get something that basically worked together in a week.

I probably should have been too ashamed of what I had created to open-source it, but thanks to my time at Google, any sort of ego I had about the quality of my code was basically gone, and I was more than happy to put it out there for everyone so that other people would be able to work on it and improve it over time. Just because I'm not the best programmer in the world doesn't mean I can't contribute useful things, and the community that has sprung up around Crunch is something I am incredibly proud to be a part of.

You wrote a blog post about building Crunch, and then having someone contribute to the open source project, as this amazing moment. Can you talk a bit more about what that was like?

It was about understanding the complicated software written, finding a non-trivial bug, and improving it. I like literature a lot. I like David Foster Wallace, and I'm wearing my favorite David Foster Wallace t-shirt. It the motto of the Enfield Tennis Academy in Latin. It translates to, "They can kill you, but the legalities of eating your corpse are quite a bit dicier."

Wallace writes a lot about loneliness. There's a character in Infinite Jest called Madame Psychosis which is a reference to metempsychosis. It is a notion of the transmigration of souls from Greek literature, that's like a John Malkovich situation of being picked up and stuck in someone else's head. Gabriel doing that fix was like metempsychosis because I put some aspect of myself into this code and he improved it. That was sublime. I was very lucky.

BRADLEY VOYTEK

Professor of Computational Neuroscience at UCSD, Former Data Evangelist at Uber

Data Science, Zombies and Academia

Brad has had an eclectic career. From working in neuroscience and academia, to becoming a world authority on zombie brains to contributing to the data science team at Uber as employee number seven, his story is one of embracing learning, overcoming challenges, and cross-pollinating ideas from disparate fields.

He is currently a professor of computational neuroscience at the University of California, San Diego (UCSD).

How did you get to where you are today?

I started as a physics major at USC in Los Angeles, and I briefly worked in an ultra-low-temperature physics lab. As a young kid, I thought that physics would be the direction I would go in. But I realized quickly after working in the lab that it wasn't really what I wanted to do.

I didn't know what to do or what my interests were going forward, but I had taken a psychology class to fulfill a general education requirement, and I became interested in the topic. Around the same time in college, I started learning to socialize better and also became more interested in other people. My grandfather, who I had grown up with, had also gotten sick with Parkinson's disease. Although he was a really smart engineer, he started to decline cognitively really quickly. The confluence of all these things that happened at the same time in my life made me realize that I needed a shift in my long-term career path, and neuroscience became something that I was interested in.

As an undergrad, I started working in a neuroscience research lab, and the very first project I was assigned was to take flat text files and copy and paste different parts into Excel to aggregate the data. They gave me two weeks to do this, and I was like, "This is ridiculous." I wrote a simple C++ script to do it for me, and I came back the next day with all of it finished. To the other people in the lab, it was like I had worked some kind of magic. Programming was this amazing thing that they did not understand.

From that point forward, I became the "tech guy". I started automating a lot of the things that were going on in the lab. I found my niche there.

After graduating, I started working at UCLA in the Brain Mapping Center, and I was the PET (positron-emission tomography) scanner operator. PET is a type of non-invasive brain imaging and I was running the PET scanner and collecting data from people. I started doing a little of my own research and used that time to figure out if I wanted to go to grad school and do my PhD. Through this experience, I realized that computational neuroscience was what I wanted to do.

I applied to places in California like UCSD, Berkeley, UCLA, and UC San Francisco. I almost didn't get an interview anywhere because my grades in undergrad were terrible, but I got lucky and got into Berkeley. It was an amazing environment with a ton of incredibly smart people. Berkeley has just now started its own data science institute, but it was clear when I was there from 2004 to 2008 that this idea of "data science: was percolating through the Bay area.

They gave me two weeks to do this, and I was like, "This is ridiculous." I wrote a simple C++ script to do it for me, and I came back the next day with all of it finished. To the other people in the lab, it was like I had worked some kind of magic.

At the end of my PhD, I was approached by my friend Curtis Chambers, who was the first head of engineering at Uber. He was employee number four in the architectural dispatch system and was a close friend of mine in high school (I was the best man at his wedding). He said, "We have a ton of data, and we don't have anyone to do anything with it. I know you do this kind of stuff. Would you be interested in working with Uber?"

At that time, I had just finished my PhD, and my initial reaction was, "I don't think it's that interesting." However, as we talked more, I started to get a better idea of the company and I decided to go meet with the CEO. I had lunch with Travis Kalanick, the Uber CEO, and he wanted me to do a coding challenge to see what I knew. I said to him, "Look, you can have me doing coding challenge games, but how about you give me your data to play with? If I haven't done something cool by the end of the day, that will settle it."

Travis liked that, and so they gave me some data to analyze. I sat around and hacked at it for a while and, by the end of the day, I had some analyses and visualizations for them. That's how my work at Uber started.

You mentioned that you applied to different graduate programs and only got

accepted into UC Berkeley, despite your low GPA. What did you do to convince them that somebody with your unorthodox background could do a PhD there?

I don't know. I wish I had a solid answer. I actually asked somebody very high up in the department how I got into Berkeley when I couldn't get in anywhere else. The professor said very bluntly, "Well, we looked over your application and thought you were a fuck up, but we thought you were a fuck up with potential so we decided to give you a shot." I don't know what the potential was. I do a lot of writing and public speaking so I think I am able to communicate ideas clearly, so that might have helped me in my application.

I think Berkeley also embraced the Silicon Valley ethos where failure is something that helps you move forward. In a lot of other places, failure is generally looked down upon, but I think there's something to the idea that failure is how you grow. I've been embracing this philosophy for a long time, and I didn't try to hide anything in my application. I said, "Here's what happened. It's not an excuse. This is just what happened. Here's the story, and here's what I learned from it." I think most people didn't really care, but every now and then, it just takes one person to actually read what you write and appreciate it.

I said to [Travis], "Look, you can have me doing coding challenge games, but how about you give me your data to play with? If I haven't done something cool by the end of the day, that will settle it."

Now that I am a professor, I just participated in my first rounds of admissions for the PhD program at UCSD. UCSD's Neuroscience department is one of the best in the world. It's a very competitive program, and during the admissions, there was one person who I wanted to admit. This person had a low GPA but had strong GREs and an incredibly strong background. They had thought very clearly to get to the point where they were. They were slightly older than the other applicants because they had done real world work instead of going to graduate school directly, so I highly recommended that this person get accepted.

If you look at other professors' CVs, listed there will be numerous publications, incredible companies they've helped fund, students they've mentored who are now amazing professors, and incredible research grants that they've received. I remember looking at those as a fresh PhD student and thinking "I'm not cut out for this." I couldn't even imagine what it took to write a single research paper and get it published. I couldn't imagine doing that once, and I saw people with over 200 publications.

In my CV, I actually have a section that's listed as my failures so for every paper that I've gotten published, I say how many times it was rejected from different journals. I list

every grant that I didn't get which I applied for, every fellowship, and every faculty job I applied for but didn't receive. Everything that I've done that didn't come true is listed in the failure section of my CV, and I've gotten positive feedback from students looking at that because it's a litany of crap. There were papers that were rejected by 10 journals before they got accepted. I think people don't recognize that behind these incredible CVs of these 60-year old professors, they've got 60 years of failures that they had to go through in order to do it. I try to be a little honest about that.

DJ Patil had this great quote that, to paraphrase, goes something like, "In the very beginning, in order to do something new, you need to leap across this chasm and you need someone on the other side to catch you in order to cross it."

It seems like from your experience in graduate school, you've really internalized that and believe in that. Now you're on the other side of the chasm, trying to catch people, hoping that they can make it across despite their unconventional training.

DJ is a smart guy and I like that analogy a lot.

I come from a pretty low socioeconomic status background. My family's not well off at all, and to get to where I am today, I can easily name at least a dozen people who gave me lucky breaks. Everybody likes to talk about the value of hard work and work ethic, but getting to where I am today required tons of luck. It required someone to reach across that chasm for me, and I don't know why. I'm certainly on the other side now, trying to do the same for as many people as I can.

What was your exposure to computer science and the idea of interacting with data? How did that evolve as you went through your undergrad research and PhD research?

In hindsight, what I did throughout that undergrad project was very simple data munging. USC didn't have a neuroscience major at the time. They had a psychology major and a neurobiology major, but I wasn't interested in neurobiology or cell molecular biology, so I tried to throw together a major on my own.

I ended up taking courses like Introduction to AI and C++ programming. I took these classes because I had several friends who were computer engineering majors and after spending a lot of time talking to these guys, I thought that programming would be a useful skill to have.

When I worked in the lab, I realized that my programming skill was applicable to the problems. For example, the lab I worked in was doing brain imaging, and the analyses that

they were doing seemed complicated, requiring the use of obscure programs. I realized at some point that it was just matrices of data. Once you had the realization that these are just numbers, it allowed me to do a lot more. You can start writing your own analyses and doing things that people may not know or understand. It's mind boggling because with two lines of Python, you can do much of data analysis much more efficiently.

I told this story once, and someone told me, "In the land of the blind, you are the one-eyed man." That's essentially all it is. You find that you have a skill set that is valuable to the field that you're working in that not everyone has. Suddenly, it's just something magical that you can do.

In 1999, I imagine that the science lab was not very technical. You coming in and applying programming must have been mind boggling to them. It was something that you've seen in class, but for people who haven't been exposed to programmatic data analysis, that must have seemed like magic. Similar to what you do now, at that time you were leading the way and evangelizing the different ways that people can think about and manipulate data.

In my CV, I actually have a section that's listed as my failures so for every paper that I've gotten published, I say how many times it was rejected from different journals. I list every grant that I didn't get which I applied for, every fellowship, every faculty job I applied for.

I remember taking a statistics class in psychology. They were trying to use SPSS which is a statistical package for social sciences. It allows you to do things like regression analyses, and ANOVA. I remember being confounded by the idea of assuming that the data followed certain probability distributions. I didn't quite understand why you're making assumptions, and I didn't get the difference between ANOVA and the t-test. Then as a grad student, I remembered that they were all the same thing. You have a general linear model; the t-test, ANOVA, and regressions are just extensions of that.

In my lab right now, I have a lab manager who is a fresh out of undergrad. We were talking about that, and I described to him how a t-test by drawing it on the board really quickly. He said, "I took a year of stats, and I never got it as clearly as I do now from what you just drew."

It's shocking to me how bad people are at explaining data, data science, and statistics. It's not a magical thing. There's a reason some people are so good at it. It takes some time internalizing and getting it, but once someone shows you that, it just becomes so clear.

It requires you to think outside of preparing for a test, and see how the ideas can become an actual tool you can use.

Right. I think my failure to become a physicist was that I was too young to get that. When I was taking physics classes, it was memorizing all these different equations and trying to figure out which equation to plug into. That didn't seem interesting to me. Now, of course, I realize it's not what physicists do. I think if I had figured that out earlier, I could see my career path diverging.

Right now, you're quite active on Quora. You teach different people about many different concepts, and I see that as being a very important package of being an effective professor or data scientist. Can you talk more about this missing aspect of data science that isn't as heralded, which is the aspect of being able to communicate effectively?

Yes. I always think back to the movie *Office Space*, which was making fun of the first dotcom industry. In the movie, there's a great line where they're trying to figure out who to keep and who to fire in this tech company. They're talking to this guy who is a product manager. But since he's a product manager, he's not a manager per se, so these guys that came in to interview him are asking, "What do you do?" He's replied, "I talk to the engineers, and I learn what they're doing. Then, I relay the information clearly up to the management." They said, "Why can't we just have engineers talk to management?" And he says, "They need a people person."

I think about that a lot. There's something very critical about being able to communicate your ideas effectively. When I came into Uber, one of the things I thought about was that, before they got bought by Match.com, OkCupid had a really good data blog. They were using all of these dating interactions on their website and metadata about people to try to do analyses about what gets people dates and what people find attractive. I read those well before I was a grad student and well before I got into data science. Everybody loved it.

When I started working with Uber, I was thinking about how the data can be used to tell an interesting story. Just like writing code, telling a story effectively takes a lot of practice. That's a part of the reason why I do a lot of writing on Quora. I teach, and I do a lot of public speaking at elementary schools, junior high schools, high schools, or at a bar to a bunch of drunken aficionados. It's practice. Just like I have to sit down and practice writing code, I also have to sit down and learn how to communicate the idea.

My wife is actually a very good sounding board. Whenever I write something, I always pass it by her because she'll read something and say, "You're making this more complicated than it needs to be. You can explain this in fewer words. You didn't connect from A to C.

You skipped over B."

I remember the first time I took a programming class. It was an algorithms course. The homework was to write an algorithm for making a sandwich in which you had to explain every step you took to make a sandwich. You realize so many parts you skip over that you think are obvious, but if you had to program a robot to do it, simple things like pulling the knife out of the drawer must be explained. You have to explain exactly how you pull the knife out of the drawer to spread the mayonnaise.

We skip over a lot of stuff that seems obvious, but it's not always obvious if you're not the person staring at that data all day long. It's a good point of practice to try to remember how to be very explicit about every step that you take and connect the dots for people.

You've worked with data while at Uber and have an academic background with a lab in UC San Diego. Do you think that your academic background better prepare you for Uber?

Absolutely. The one thing that you get from an academic PhD in data science is learning to tackle a big problem by breaking it down into bite-sized chunks. When you start a PhD, what you're doing is saying, "I am entering this 3000-year-old human endeavor of trying to figure out where we are in the world and what we're doing, and I think I can add something new." That's a ridiculous assumption, but people do it head on.

It's shocking to me how bad people are at explaining data, data science, and statistics. It's not a magical thing. There's a reason some people are so good at it. It takes some time internalizing and getting it, but once someone shows you that, it just becomes so clear.

You start reading papers, and you know what you're interested in. You see that there's a hole somewhere, that there's something missing. You think to yourself, "I can add something. How do I go about tackling that, addressing that problem? How do I define the scope of the problem, and what do I need to do to tackle it?" That's what you're trained to do as a PhD student and is the important skill that you don't get if you skip academia and go directly to data.

If you are fresh out of your undergrad and go work in a company like Facebook as a data scientist, you've got access to two billion people's worth of data. Unless you had an amazing undergrad experience, you don't know how to begin to tackle that. How do you wrap your head around what kinds of problems to ask, and once you have the problem in mind, how do you tackle it? What do you do?

One of the things that struck me was that people in research tend to de-emphasize the infrastructure required to do large-scale analysis. That's not as sexy as being able to ask the research questions, but it gives you more breadth. You can ask bigger questions if you know how to leverage the industrial tools.

Yes. It's helped on every level. When I started in Uber, there were seven of us and we were working in a co-working space with a bunch of other startups. The hustle of a startup is something that a lot of large research institutions don't have.

In my lab, I do a lot of methods development for analyzing human brain data, but previously I did that in MATLAB and posted the MATLAB file on my website and linked to it in published papers. It's complicated and doesn't really make sense. Now, I'm converting everything so the code in the notebooks are also tutorials. These are things I'm developing in my lab and growing in the lab culture.

A common problem that's endemic to academic researchers is if some PhD students or post-doctorates do a really good project but have the data backed up on their computer. Then, if they leave or graduate, the data is gone. This happens at least 75% of the time. These are things that regress in the development of my research lab.

On the actual research side of things, I have learned different ways of looking at data. There's a company called Lumosity. They do online brain games, which is interesting, but the thing that really interests me is that they have more data on human cognition than have ever been collected in the history of science.

We looked at Lumosity data that measures your distractibility. I looked at the distractibility average across geolocation areas like California, New Mexico, or Washington. You just have these aggregate values, and I pulled up these data that estimate country or state-level IQ and GDP. I found that state-by-state and country-by-country estimates of distractibility correlate with fatal car accidents. Countries or states where people are more distracted are more likely to have fatal car accidents, which makes sense. It's statistically robust. It doesn't appear to be an outlier. Ultimately I'd like to publish all the scripts I used to go from raw data to final published figures my research so that anyone who reads the papers can do the same analysis, or build on it.

You're one of the few people that we've spoken with who have gone back from industry to academia, and it still sounds like you're playing with private data sets of different companies like Lumosity and Uber. Why did you not choose to stay in industry? What advice would you have for these PhD students who will read our book who are trying to leave the more standard track of become a professor for industry?

Getting a faculty job is highly unlikely. Ten or 20% of PhD students will go on to get a tenure-track job, so in my lab, I'm trying to train people to be able to do other things beyond research. We're doing the research, but we're also trying to train them in version control, Python, and data analysis to make sure they have a skill set that is transferable if they decide not to go into academia.

When I started working with Uber, I was thinking about how the data can be used to tell an interesting story. Just like writing code, telling a story effectively takes a lot of practice.

As for me, not staying in Uber and going into academia was a really hard call. They offered me a full-time position and a lot of stock. That was before the stock is where it is right now. I was doing growth projections early on, and I knew where the company was going. What made it hard was that while Uber was doing some cool work, at the end of the day neuroscience is where my heart is. I try to refrain from using the word "passion" because people have a misunderstanding of passion, but neuroscience what I really want to be doing. It's more fun and exciting for me at the end of the day.

Another big part is that I've gotten a lot of big breaks over the years, much of which has come through really good professors. I wanted to give some of that back, and academia is one way to do that.

For example, I'm teaching an introductory class on data science at UCSD for the cognitive neuroscience group.

Students don't understand why they have to do computations, and I'm trying to explain to them that right now part of the criteria of understanding cognition and intelligence is data. Part of it is my desire to give back through teaching. Part of it is that the research I'm doing has a lot of big long-term potential plans in terms of public health. And I have to say that it feels good.

That's a really powerful and amazing message to share with people who are going to be interested in your background, that you chose to stay in academia. The final thing I want to talk about is what is up with you and your interest in zombies? What's up with that?

Haha, are you asking what's wrong with their brains? Actually this has been surprising and how much it's taken off. This goes back to the science, communication, and outreach. If I go to a high school and talk about how neuronal shot noise and channel leakage leads to an increase in neural noise, or how to estimate areas of communication between brain

regions, people's eyes would glaze over and they'd say, "What are you talking about?" High school students don't care, and quite frankly, most people don't care either.

If I go back to high school and start talking about zombie brains, show videos of zombies eating people, and explain why zombies are doing that, suddenly people are going to understand. It's Trojan-horse teaching.

However, if I go back to high school and start talking about zombie brains, show videos of zombies eating people, and explain why zombies are doing that, suddenly people are going to understand. It's Trojan-horse teaching. It's a gimmick that we used to do science communication and outreach, and it took off. People are really into it. I was on some National Geographic TV show; Princeton University Press recently published our book, *Do Zombies Dream of Undead Sheep?*, about the zombie brain soon. We got to meet George Romero, the director of the original zombie movie, *Night of the Living Dead*. There are tons of speaking engagements. I was a guest of honor for a science fiction and fantasy conference in 2014.

It's a weird shadow career I have, but it's been a ton of fun. It's really silly and it works. People seem to dig it. It's nice. It's powerful and effective communication.

Out of all the things you've done in your life, the theme here is your willingness to write, talk, and teach people. That's also one of the reasons we're grateful that you took time to share with us your stories.

Thank you for being so honest and willing to share both the stories of success, and failure and struggle. I think a lot of these stories are going to resonate deeply with the readers, especially those who branch out from the traditional path.

Thanks, guys.

LUIS SANCHEZ Founder / Data Scientist at ttwick

Academia, Quantitative Finance and Entrepreneurship as a Data Scientist

Luis trained as a civil engineer in Venezuela before arriving in the US for his MBA on a Fulbright in the early 90s. Though he aspired to join the World Bank, Luis found an alternative application of his data skills in the world of finance.

After an illustrious career as a quant at AIG and Deutsche Bank, Luis found himself structuring exotic asset backed and catastrophe linked securities at Lehman Brothers (you can see his video here), where he worked till the firm's bankruptcy on September 15, 2008. With plenty of free time on his hands, he dabbled in cross pollinating the ideas from his structuring days to areas of social media.

He is the founder of and data scientist at ttwick, a search engine for social media content.

Where do you work?

I am the CEO and Data Scientist of ttwick Inc, a data analytics startup with roots on Wall Street.

What was your personal career path like?

I started my analytical career as a Civil Engineer specializing in structural engineering, hydraulics and numerical analysis with a degree from a military university in Venezuela in 1987. I also obtained a specialization in systems analysis and programming from another Venezuelan institution, but it was the combination of engineering and programming that put me in the right frame of mind and gave me the skills to eventually evolve into a data scientist.

In 1990, I decided to move to Washington, D.C. and begin studying for my MBA on a LASPAU scholarship (a Fulbright scholarship for Foreign Students). My goal at that time was to get accepted into the World Bank's Young Professionals Program and work in global infrastructure development. I dreamed about all the data I would have access to at the World Bank, to play with and analyze.

Back in 1990, it was not easy to get enough data to analyze, and I used to spend a lot of hours in the computer lab with my newly issued email address and Internet access. I

started to gather as much data as I could via Gopher, Archie (the first search engine for FTP archives), or whatever I could get my hands on. I then discovered the University of Chicago's CRSP database, which had some securities data available in a monthly format. In the summer of 1991, I got a 2400 bps modem, which allowed me to access the CRSP databases from my IBM PS/2 at home, and finally, I did not have to spend so much time in the computer lab, which always annoyed Gabrielle, my girlfriend at the time. In 1991, it was difficult to obtain large volumes of high-quality data, so I started to explore other methods such as creating synthetic data via Monte Carlo simulation, which kept some of the features of the original data set. Because of that, I started to do a lot of research in computer science.

I never got the job I wanted with the World Bank, but after graduating in 1993 I started working for a New York-based hedge fund that wanted a quant for its newly established Quantitative Analysis department, to complement the work of analysts doing pure fundamental analysis on securities.

By the way, I ended up proposing to Gabrielle, believe it or not, after writing a program based on professor Thomas Saaty's Analytical Network Process. I thought his multiple-criteria decision algorithms would help me make sure I did not have any inconsistencies in my logic. (Years later, Gabrielle got kind of upset when she learned I based my proposal decision on an algorithm.)

What are your responsibilities at ttwick?

I found out a lot of the market-based algorithms for allocation of resources I developed and used in finance could also be used to search and organize unstructured data on the web; to determine if an online ad should be launched today or in a few days; to create real-time portfolios of content at low cost; to dynamically calculate probabilities of reaching certain audiences, etc. These algorithms coupled with natural language processing, data augmentation, and other techniques could be used for many applications, including the discovery of arbitrage opportunities in certain financial markets, forecasting of political events, etc.

As CEO and Data Scientist, I am developing a series of B2B and B2C products and consulting services based on the above-mentioned applications, some of which are currently being used and/or tested by hedge funds, advertising agencies and other institutions.

What is data science to you?

To me, data science is the art and science of extracting actionable intelligence from sets of data, big or small.

I call it "art" because there is not really a one-size-fits-all technique that can help you answer the questions you want to ask your data. You need to be creative and have imagination to see what others don't see in the data. If you are anything like me, the best solution to your most challenging problems has come to you when you least expected it, in the form of an inspiration. When that happens, I get in the zone and the solution just comes to me, and I can't focus on anything else.

Data science is the art and science of extracting actionable intelligence from sets of data, big or small.

I call it "science" because you need to know the theory behind what you do and put in your 10,000 hours of problem-solving so you develop "muscle memory" so to speak, and you acquire the right foundation to become a good data scientist.

One thing I believe but don't know if other people would agree with: Good data science can't be 100% theoretical or 100% practical. There has to be a mix.

So, in your opinion, what is the goal and purpose of a data scientist?

The purpose of a DS is to extract actionable intelligence from sets of data with the most efficient use of resources and under time constraints. The DS should be able to connect data together in meaningful ways, to create new knowledge from the combination of data, to be able to analogize and solve problems in creative ways, and to do all that quickly. Like General Patton used to say: *"A good solution applied with vigor now is better than a perfect solution applied ten minutes later."*

What are some past projects you have worked on?

As a financial quant, I worked on many interesting projects, many of them being the first ever of their kind. Many of the projects on which I worked set the foundations for niche markets within structured finance and trading.

Some of the most interesting deals were:

- **First sovereign catastrophic (CAT) bond:** Using a parametric structure, I designed an ABS structure that covered the Government of Mexico against the effects of an earthquake. The bond was rated and successfully launched to market.
- **Weather options:** one of the most fascinating projects during my time at AIG, where I was in charge of developing a model to price the risk of extreme rainfall or temperature in several cities across the world. This topic is extensive, but developing

the mechanism for a new market and seeing it grow and find participants that included energy companies, theme parks, airlines, etc., was a fascinating experience.

- **Music and film royalties:** Have you heard of Bowie Bonds (as in David Bowie)? Basically, the same idea was involved in deals I was pitching while at Deutsche Bank for some high profile artists. After I left in 2005, I retrofitted my own model to value film productions yet to be made, and all of a sudden, I found myself giving speeches at Hollywood events about the use of Monte Carlo Simulations and Bayesian analysis to price film productions.

More recently, as a DS, I demonstrated to a hedge fund that there is a way to gain an edge in the market via non-conventional financial analysis, using non-conventional sources of data and machine learning tools such as those I am developing at ttwick. This sort of analysis, coupled with the right financial derivatives, could generate superior alpha returns in various market environments.

What was your experience like transitioning from academia to industry data science?

Good data science can't be 100% theoretical or 100% practical. There has to be a mix.

The experience was effortless, with a dash of good timing. As I mentioned before, I was applying the math and coding skills I had acquired as an engineer to solve problems presented as part of the curriculum of my MBA. I found it interesting that only a few of my classmates had any coding skills, so I became the "class quant." Classmates came to me for help with assignments for classes such as Strategic Marketing, where data science could be applied to solve marketing problems. By helping them, I also gained exposure to their areas of concentration.

I started to write code for small projects with demos of things for marketing, financial trading, etc., and towards the end of my MBA, I started to attend specialized seminars in New York that concentrated on real-life problems about trading non-liquid securities, correlation trades, technical analysis, etc.

Over several of these seminars, I started to see a core group of people attending the most interesting ones, whom I later learned were investment managers looking to expand their knowledge of the markets, while at the same time scouting talent for their own hedge funds. I met Marc Chaikin, a famous technical analyst who created one of the first platforms for real-time technical and fundamental analysis of securities, with a huge database of tick-by-tick data.

Eventually, I ended up working for Marc's best friend, a smart hedge fund manager, Chris

Castroviejo, who had left Bear Stearns and teamed up with a group of traders and analysts who were starting their own group of onshore and offshore hedge funds.

My last job in my over 15-year career as a Wall Street financial engineer was Senior Vice President for Lehman Brothers, where I was a Senior Quant.

What drew you from Financial Quantitative Analysis to Data Science?

It was very simple: after the bankruptcy of Lehman Brothers where I was a Senior Vice President, I (as well as many other quants), was in "job limbo," meaning there were practically no jobs available anywhere, much less for structurers of exotic assets. I transitioned to Barclays Capital, but by the end of the first quarter of 2009 I was laid off along with most of the senior structurers and bankers who made the transition from Lehman.

The general public barely understood what a credit default swap or a credit transition matrix was, much less credit-linked notes, synthetic CDOs, CAT bonds, Markov-chain Monte Carlo and some of the instruments and techniques I had specialized in over the years. There was no appetite for any sort of exotic instrument, and no capital available for anything.

It was very frustrating because precisely then, as a consequence of the credit crunch, some of the best investment opportunities I had ever seen in my life materialized in the form of "Obama projects," but I did not have any funds to invest, and they were clearly advantageous only to the wealthy with knowledge and some political connections.

Along with a couple of my friends from Deutsche Bank and Lehman Brothers, I put together a basic infrastructure to raise capital and invest in such opportunities created by the Obama administration, but we were not able to get any commitments by the deadlines set by the programs, and I found myself with a lot of free time, compared to the 70-90 hours per week I was used to putting in at investment banks.

At the same time this was happening, I found myself improving my programming skills, and learning all I could about data scraping, web crawlers, and artificial intelligence. I was fascinated by the possibilities. I started experimenting with spiders to crawl some sites, and all of a sudden I was using the data in creative ways to find solutions to real-life problems.

One of those problems had to do with valuation of film productions. Years earlier, I had created a rather sophisticated valuation model, focused on securitization of film assets for Hollywood clients. I wondered if, by using comments in social sites, I could improve

the accuracy of the model. I succeeded, but first had to develop additional algorithms to filter out the noise.

I kept coding and discovering more interesting things about the film industry, but was particularly attracted to the Internet Information Provider subsector of the tech market, since I could see analogies in the way Hollywood earns profits from films: you need to be in content creation, distribution, and advertisement in order to minimize your risks.

So I started reading the 10Ks and 10Qs of Google, Yahoo and other companies in the sector, and doing Monte Carlo simulations to learn about their strengths and weaknesses. I learned a great deal from that exercise. At that point, I started considering myself a hybrid data scientist/financial quant, which I believe is a rare combination.

What would you have done differently if you were able to speak to yourself right at the end/middle of your graduate school career?

If I had that opportunity, I think I would just tell my earlier self to code more in languages other than Visual Basic. I would set up a plan for my earlier self to learn Octave and Python, and Java when it came out in 1995.

Focus more on what your own strengths are and less on what is perceived as a cool career path at the time.

If I had the opportunity to speak to myself right around the middle of my professional career, let's say circa 2001, I would have told myself to stay in New York as a general quant, instead of moving to London to be a front office quant pitching deals to European governments and corporations. Not that I didn't do a good job as a structurer and relationships guy, but an unintended series of events happened around that time which I think could have had a better outcome for me and others, but who knows?

My advice is, focus more on what your own strengths are and less on what is perceived as a cool career path at the time.

How do you describe the value that you and other DS bring to the company?

The value I bring to ttwick is my diversified analytical background and real-life experience in several industries that helps differentiate ttwick from other startups operating in DS.

I have been conducting informal job interviews with PhDs I might eventually hire from diverse fields such as physics, economics, linguistics, engineering, microbiology, etc. with amazing potential to become Data Scientists.That is the sort of analytical diversity I want to have at ttwick.

What are some surprising things about working in industry vs. academia?

I can best answer that question by giving two examples of situations taken from my own career.

The first example is the design of a financial structure to cover the government of El Salvador against the effects of excessive rainfall, which has a definitive impact on the GDP of that country. I was given that task in 1998 or 1999, I can't remember. El Salvador's economy is mainly driven by agricultural production, and extreme weather events could have catastrophic effects on the economy. Conventional countrywide insurance protection was either too expensive or unavailable, so I started to design a parametric bond-type structure. As usual, data was of the essence.

I got the data, but it was terrible, with wild swings in it that could not easily be fixed. The time-series had a numeric code representing the station, the name of the town/village, and the daily precipitation. The gaps represented a big problem in valuing the risk. But then I did some interesting data exploration: I asked some of my younger colleagues (Associates and Analysts) to help me gather the geolocations of the stations.

I shared the findings with a government official in El Salvador who confirmed that the time periods of wild swings or gaps in the data corresponded more or less to the heights of internal political conflicts in El Salvador.

The time gaps corresponded to more or less well-defined, circular areas, always near a river or stream. I shared the findings with a government official in El Salvador who confirmed that the time periods of wild swings or gaps in the data corresponded more or less to the heights of internal political conflicts in El Salvador. The geographic areas I showed them represented the geolocations of what later were found to be the main camps of the FMLN (a guerilla coalition at war with the government in the 1980s and '90s). By looking at the wild standard deviations and their locations in time and space, you could more or less predict the next hideout of the guerilla leaders, since they seemed to follow a defined pattern. It turned out the fluctuations were simply due to the destruction or misuse of pluviometers by the guerrillas, coupled with the fact that data collection was not frequent. In any case, I came up with a solution for the problem of rainfall measurement at certain points by correlating rain accumulation with water levels of rivers near the data collection stations, where I had better data.

The second example was from my time at Lehman Brothers. I was analyzing a corporate deal in an emerging market that involved a commodity as collateral. I had the data I needed for a conventional type of analysis. But something did not feel right , so I decided to enlist the help of another colleague, Jami Miscik, also a Senior Vice President.

Before Lehman, Jami ran the National Clandestine Services of the Central Intelligence Agency under George Tenet. Around 2005, she joined Lehman Brothers as Global Head of Sovereign Risk Analysis, and to this day I consider her a very unique type of data scientist.

I was fascinated with Jami's analytical skills for political risk; she's equally talented in the art and science parts of data science. While at the CIA, Jami ran a complex quantitative and qualitative program to forecast political instability in 40 countries based on 25 indicators, and I was lucky enough to be one of the few executives invited to gather for her weekly world outlook meetings. At the suggestion of the Head of High Yield Trading at Lehman, I enlisted Jami and her team to dig a little deeper on the company I was analyzing.

After her report, I decided not to go ahead with the deal (this could be a topic for another book), and informed Jami of my decision, but in any case, Lehman was already in bad shape. I called to set up a meeting with her to personally thank her for her great work and see if I could discover a little more about her proprietary models and exchange ideas on a couple of topics. Both her calendar and mine were free for the same day: September 12, 2008, which turned out to be the historic last day of operations of Lehman Brothers. Nevertheless, we still kept our meeting and managed to talk for a little while.

A couple of years later, I started thinking about creating real-time political risk indicators by tapping and correlating many sources of publicly available data on the Internet, and "calibrating to market." We did it and now we are successfully using our analysis to advise a few interesting groups out there..

I think the likelihood of getting exposure to the sort of data problems described above is zero or very small if you only have exposure to academia.

How do you measure your own personal career success?

A few years ago, I realized I had always been an entrepreneur inside much larger organizations and managed to obtain R&D funding to develop and launch securities that many times were called crazy or impossible to launch.

So I ended up as a DS because when I started to analyze what my skills were, I realized that a DS is more or less a financial quant minus the financial knowledge. Remember, financial quants come from all fields: computer science, physics, math, economics, finance, etc.

At this moment, I measure my own personal career success by the fact that ttwick exists,

it is funded, and some of my investors are either former bosses or colleagues of mine, and even traders, analysts, and/or executives of well-known financial institutions and media companies.

How do you work with other people on your team?

The best data scientists probably have experience solving problems for diverse industries.

I assign tasks to the teams with the most relevant experience to the problem at hand whenever possible. Then I have periodic progress reports to review their work, and if I don't understand something, or think there might be a better way to do it, I encourage discussion. We have all gained interesting insights from this process, and so far I am satisfied with the results.

I have several machine-learning specialists, data wranglers and general coders on my team who support what I do in DS. I would like to hire more DSs like me, but it is very difficult to find "diversified" DS, and even more difficult to find any with financial backgrounds.

Everything we're doing at ttwick involves multiple disciplines, and my experience with multidisciplinary teams across several industries allows me to have fluid communication with my team members.

What type of careers could people working in DS transition into if they decide DS isn't for them?

Based on my own experience, I would say that financial quantitative analysis would be an option, but a person would have to learn at least some basic finance and get a CFA (or better yet, a CAIA) if they want to excel.

What separates the best data scientist from the merely good?

Since I describe data science as the art and science of extracting actionable intelligence from data, I would say a good data scientist has an academic and professional background that makes him or her good in the science part, but the best data scientists are talented in the art part as well.

What backgrounds do these best data scientists tend to have?

I can't really say, but to me, more than having a particular field of concentration, the best data scientists probably have experience solving problems for diverse industries.

That creates the right frame of mind to attack problems from different perspectives; this might not be obvious to somebody who has only worked in one industry.

Who are some DS people you admire? Who are people doing interesting things with data that you admire?

To me, Hilary Mason is a great data scientist whom I would love to meet someday (and maybe invite for a cheeseburger). I find her work very interesting, as well as the work Claudia Pelrich at Dstillery has done in the area of detecting fraud in online advertising.

Also, the work that professors David Cope, Larry Polansky, Peter Elsea and Daniel Brown at the University of California Santa Cruz are doing with artificial intelligence applied to creativity is extremely interesting. I spent a few weeks interacting with them at the UCSC, and what I learned is truly fascinating; for example, the use of non-speech audio to perceptualize data as a complement to visualization techniques, or even as a stand-alone technique.

There is also a group of Wall Street data scientists (or quants) whom not many people know of outside of trading, structured finance, risk management and/or political risk analysis. I already mentioned Jami Miscik and Marc Chaikin, but I should also mention Jorge Calderon and Phil Weingord (ex-heads of Global Asset Securitization for Deutsche Bank and Credit Suisse), Dr. Mark Shi at Citigroup, and Dr. Jose Hernandez and Blythe Masters at JP Morgan. I have worked with all of them, with the exception of Blythe Masters, who almost hired me for her Credit Risk Trading group, but I opted to take another offer.

The group above created many cutting-edge analytical and computational methods used today in financial engineering, risk management, and other fields, but unfortunately, the 2008 financial crisis and problems in the risk valuation of many MBS and CDS by rating agencies tainted a big chunk of the industry and overshadowed the work of a lot of good people.

Can you tell me about some of the ways you and other data scientists at ttwick keep ahead of the curve, education wise?

I attend a lot of conferences and meetups, and I read as many books as possible about the latest findings in artificial intelligence,, financial engineering and many other related topics.

I attended the Strata Conference 2014 in Santa Clara and it was fascinating to learn about some of the cutting edge initiatives at DARPA in the area of big data, the latest ML

R&D projects at Stanford, and in general, getting a feeling for the general direction of the industry. I highly recommend it to aspiring data scientists.

I also run a meetup group in New York called "Algorithmic Art - Quadrivium", where Iexplore Machine Learning applied to creative arts, and the legal ramifications of proprietary artificial intelligence code creating content.

What are your personal 1 and 3-year goals as a DS?

My goals are to complete all the pending projects I have for ttwick, file patents for all the technologies we are developing, and help improve the efficiency of a couple of industries in which I have not seen significant improvements in decades.

How do you think DS is changing over the next few years?

I expect the biggest advancements to come from high-performance computing and data storage.

Definitely, there will be more "toolkits" available to do data analysis. Data wrangling will become easier and faster (hopefully), data collection systems and sensors will probably have built-in data cleaning capabilities, or something like that, and testing of different methods will become faster. Also, I think data scientists will be incorporating a lot more hardcore time series analysis into their work, which I don't see much outside of finance.

What are some new developments in the field that you are really excited about?

There are several exciting new ones:

- The DeepDive probability inference methodologies that Christopher Ré is building at Stanford;
- DARPA's Memex project;
- A few of the developments related to semantic search, coupled with natural language, and the intersection with fintech, including the one in development at ttwick;
- The work of some people I know with cryptocurrencies;
- Artificial intelligence applied to algorithmic art (music derivatives, visual arts, etc.)

I think any progress in any of those initiatives, along with adequate user adoption and new ethical business models for the data economy — all of these have the potential to disrupt many industries over the years to come.

MICHELANGELO D'AGOSTINO

Senior Data Scientist at Civis Analytics

The U.S. Presidential Elections as a Physical Science

As an undergrad at Harvard, Michelangelo was fascinated by physics. He finished his PhD in astrophysics from Berkeley, and developed a love of working collaboratively on hard problems with other people while analyzing neutrino data for the IceCube experiment.

Michelangelo started his data science journey as a senior analyst in digital analytics with the 2012 Obama re-election campaign, where he assisted in optimizing the campaign's email fundraising efforts and analyzed social media data. Afterwards, he worked as Lead Data Scientist at Braintree before he rejoined many of his former colleagues from the Obama reelection team at Civis Analytics. At Civis Analytics, Michelangelo works on statistical models and writes software for data analysis.

Michelangelo has travelled to the South Pole and has written about science and technology for The Economist.

In his interview, Michelangelo shares his story and offers practical advice on transitioning from a PhD into data science. He also shares his thoughts on data science for social good.

Can you talk about your career from undergrad to PhD? How did you transition to data science and data analysis, and what were your various roles afterwards?

My career has been strange. Sometimes, it feels like a random walk, but it's more of a greedy algorithm. Every time I've had a choice of what to do, I think about what seems like the best opportunity, the most interesting thing for me to do. It's worked out really well even though there hasn't been this overarching plan.

I started as a Harvard undergrad and studied physics. I always really loved physics, but I also really loved doing other things outside of physics. So, I took tons of literature and history classes when I was an undergrad. I liked working in the lab and doing the research stuff, but I always had lots of different interests.

I graduated, and I wasn't sure if I wanted to go to grad school because I wanted to get a job. Looking back on it now, I wish data science existed when I was an undergrad. I really

loved quantitative research. I loved the stuff I did in the lab, but it felt a little distant to me. It didn't feel connected to the world. I think that's what always made me a little hesitant about research, but when I graduated from college, there was not really a path for technical people to do things outside of academia. You could go work in finance, and tons of people I know worked on Wall Street. But outside of that, there wasn't a clear thing to do.

My career has been strange. Sometimes, it feels like a random walk, but it's more of a greedy algorithm.

I took a fellowship to go teach physics at a boarding school in England for a year because I also really loved teaching. It was a great way to experience teaching physics, learn about high school kids and what they're like, and travel around Europe. I really enjoyed it, and I could really see myself teaching for a long time, but I started to apply to grad school because I knew that teaching would always be there. I liked it and I knew I could go do that. I also knew that if I wanted to go back to grad school, I felt like I had to do that relatively quickly before I got too old and just too tired.

I started grad school at UC Berkeley in physics, and I enjoyed the classroom aspect of it. I started doing research in condensed matter physics. I enjoyed that, too, but I was basically in a second sub-basement. I was doing this condensed matter research, and I had this feeling that it was detached from the outside world. Also, it was really solitary.

I made a transition and switched research fields to particle physics and astrophysics. I did my PhD on a neutrino physics experiment that is located at the South Pole. It's called *IceCube*, and it's operating now. We basically buried sensors in the polar ice cap to measure cosmic neutrinos. It was a transition for me because, all of a sudden, I was working with a couple hundred people all around the world. Half of them were in Europe, the other half in the US spread out across all of these different time zones. It felt like I wasn't working on something by myself. I was working on really interesting problems with other smart people and doing really hard work. I think that was what kept me in grad school — knowing that I was working with other smart people in a collaborative environment.

I found out that it suited my personality a lot better than solitary research, and that was when I was introduced to everything I know about data science. That's when I learned most of the statistical techniques and most of the computer programming I know, and it was when I started using machine learning techniques. Basically, the common thing in particle physics now is you have a big detector, and there are tons of things happening in your detector, the vast majority of which you don't care about and are not trying to study. But you also have something happening in your detector that you care about.

The whole game is trying to figure out what is signal and what is background. These detectors are so complicated, and the signal-to-noise ratios are so low, that techniques from computer science and machine learning have really infiltrated physics. It's interesting because the old school professors don't like machine learning. You go to these seminars with old guys from the 1960s, and they ask aggressive questions and give you these looks. They don't like these techniques. They just think they're black boxes. But for the younger generations, they are common tools to do the most sensitive analysis of the thing you care about.

I started learning R and just took any chance I could get to learn — like going to meet-ups and other things that one does to learn these things out of the classroom, messing with data sets and going to hackathons.

That was how I was introduced to machine learning. For my thesis research, I used a lot of neural networks to do pattern recognition for a particular kind of neutrino signal in the detector that we cared about. I found that I liked programming and statistical work and machine learning a lot more than I liked lab research.

That was how I was introduced to this field, and I finished my PhD. I did a post-doctorate for a year in neutrino physics, and this was when the term data science first came out and people started talking about it. I started reading lots of blogs about the field, realizing that this is something I wanted to do and had the right skills for.

I started messing around with Kaggle when Kaggle first came around. I started learning R and just took any chance I could get to learn. I went to meet-ups and other things that one does to learn these things out of the classroom, started messing with data sets and going to hackathons.

One day, as a post-doc, I was in my office randomly reading KDNuggets, which is a blog for learning data science stuff. They posted an ad for the Obama campaign looking for scientists, statisticians, and computer scientists to go work for the campaign. It seemed like an intriguing opportunity for me. I had never worked in politics before, but I had always been interested in it. Because I had been reading a lot about data science and making that transition, it seemed like a good opportunity to test out if I was any good. As it was only a year, this work also seemed like a low pressure way to test out if data science was interesting to me. I didn't have to quit my post-doc. But in reality it was actually the opposite of a low pressure environment.

I applied. I had an interview, and I got an offer. The funny thing was I assumed that since it was a political campaign, they would pay me so little money that there was no chance

I would be able to accept the job. It turned out that it was basically the same as my post-doc salary, which shows you how well-paid academics are.

The funny thing was I assumed that since it was a political campaign, they would pay me so little money that there was no chance I would be able to accept the job. It turned out that it was basically the same as my post-doc salary, which shows you how well-paid academics are.

I took the job. I started in November 2011 and worked through election day. It was a transformative experience for me in a couple of ways. First, I realized that a lot of the things I was doing were not dissimilar to the things I was doing in physics. I spent tons and tons of time writing Python code to grab data from APIs (Application Programming Interfaces - the way one programmatically interacts with another application or data stream) or to scrape data. It was a lot like writing data acquisition code in physics. I was doing statistical stuff in R rather than the packages we used in high energy physics, but I was still building statistical models, predictive models. Instead of particle physics models, I was building models to predict how much money a fundraising email was going to make from its early returns. The question we had was: "If we sent an email asking people to drive to a neighboring state to canvas, who found the people most likely to respond favorably to that email?" That information allowed us to focus our targeting efforts.

I realized that the techniques of working with data, understanding statistics and being able to visualize something and tell a story about it - these were precisely the skills I learned in physics and carried very well over to the data science context.

We can talk more later about the campaign if you're interested, but we did lots of modeling, randomized experiments, e-mail fundraising optimization. It was an amazing experience. It was actually the first time I felt the technical skills I had could be used to affect the world, to work towards something that could affect the world positively. That was really cool.

Then, I thought briefly about going back to finish my post-doc afterwards, but I decided that I really liked working in data science more than I liked working in research. It was like the feeling I had when I switched to astrophysics. I like working with people a lot more than I like working by myself. I like to work on things that have more impact. You see a lot more of it in industry, in data science, than you do in research. I like the pace a lot more. I think research can often be very slow, especially particle physics. It takes 10 years to build an experiment now. You have to have a monastic personality to be a physicist nowadays.

I found the pace suited me better, and the work was actually just as interesting or more interesting to me than a lot of the stuff I was doing in physics. That's how I ended up where I am. After the campaign, I went to a startup in Chicago called Braintree, which does credit card processing for other startups like Uber, Airbnb, Github, and a lot of other growing tech companies. I started the data team there, and it was a really interesting introduction to the world of startups. Then, Braintree ended up getting acquired by Paypal in the fall, and for reasons mostly unrelated to that, I decided to make a switch. I went to work with some old campaign colleagues at a startup called Civis Analytics, that spun out of the analytics shop on the Obama campaign.

At Civis we're doing really interesting data work for a lot of political clients and campaigns like we did on the Obama campaign, but we're also working with some interesting non-profit and corporate clients. We're really trying to do a lot of individual level predictive stuff like we did on the campaign, focused on political and social good work.

That's my story in a nutshell.

You mentioned that some of the most useful things you did during your time as a PhD were working on hackathons or working on Kaggle or data sets and working with people. Do you have more to add to that? What was the most useful part of being a post-doc and PhD student for your later data analysis/data science career?

I always tell students that I think the most useful skill you learn in grad school is how to teach yourself stuff and how to figure out things that you don't know. That's one thing. The second thing is to be stubborn and beat your head on a problem until you make progress. It's really those two things.

I always tell students that I think the most useful skill you learn in grad school is how to teach yourself stuff and how to figure out things that you don't know. That's one thing. The second thing is to be stubborn and beat your head on a problem until you make progress. It's really those two things.

I feel like grad school gave me confidence. Physicists tend to be a pretty arrogant bunch. They think they can learn anything, but that was the lesson I learned in grad school. I don't know every programming language in the world, but I'm confident that if I spend a few months, I could pick up a new programming language or pick up some new infrastructure tool or modeling technique. I can teach myself those things. I can go out there and read academic papers, read software manuals, and teach myself the tools I need to get the job done. I think that's pretty common across grad school fields. Most of

the things you learn you don't learn in the classroom. You learn by completing a project and teaching yourself things. In data science, that's a crucial skill because it's a quickly growing field and it encompasses a ton of things. You can't finish a degree and know all the things you need to know to be a data scientist. You have to be willing to constantly teach yourself new techniques.

That was one of the things I learned in grad school. The other is the ability to work on a hard problem for a long time and figure out how to push through and not be frustrated when something doesn't work, because things just don't work most of the time. You just have to keep trying and keep having faith that you can get a project to work in the end. Even if you try many, many things that don't work, you can find all the bugs, all the mistakes in your reasoning and logic and push through to a working solution in the end.

And the final thing is that it really helps to have experience working with data.

Having confidence in yourself is another thing. I think that working on a really hard problem like in grad school can help you learn that. And then there are just the technical things like learning how to program, running on large computer clusters. On top of mastering those techniques, the advice I give to grad students is: if you feel like you want to leave grad school and do something else, keep that in mind when picking which tools and techniques you use for a dissertation. If you can write your dissertation in Python rather than some obscure language like FORTRAN, it's probably going to be better for you. Try to be as marketable as possible with the things you learn when you're doing your PhD.

And the final thing is that it really helps to have experience working with data. The only way to learn how to work with data is to actually work with data. You can read about it, and people can teach you techniques, but until you've actually dealt with a nasty data set that has a formatting issue or other problems, you don't really appreciate what it's like. There is no substitute for the experience of having to merge a bunch of data sets together or make a bunch of graphs to sanity check something. Or finding all of a sudden that nothing makes sense in your distributions, and you have to figure out what's going on. Having those experiences makes you a better data scientist.

So far, you've given a lot of advice for graduate students, for example working with more common tools or working with data. Can you expand on that because you are a physicist-turned-data-scientist? What is your advice for other physics PhD students or other physicists who are transitioning to data science?

My advice is to recognize the skills that you have. In terms of actually mechanically

making that transition, there are lots of ways people can learn more about the field and demonstrate their interest. From a hiring perspective, when I talk to PhD students who say they want to be data scientists, I become skeptical if they haven't taken any active steps. "Hey, I participated in these Coursera courses or these Kaggle competitions. I've gone to the Open Government Meetup and have done these data visualizations." Things like that demonstrate that you can work on problems outside your academic specialty, and they show that you really have initiative. They also show that you can teach yourself new things.

The worst thing is when people present the physics job market as terrible, and they say that's why they want to get a data science job. You don't want to hire someone like that. You want to hire someone who's going into data science because of what it's like, because they want to work on data in the real world. You want it to be a positive thing rather than a negative reason that they're leaving physics.

I'm excited about future applications where data science is going to be seen as a positive force.

To be honest, it's not a terrible reason to want to leave academia because there's no job, or because you're lonely, because you're working on a tiny, tiny problem. Those are good reasons to leave academia. From a practical standpoint though, when you're presenting yourself to other people, I think you should focus on the positive reasons that you're excited to do something else rather than negative thoughts about about what you're doing. Having said that, all those things are true, and all those things are reasons why I also personally decided to leave.

The other piece of advice I always give to job seekers is when people talk about data science jobs, it can mean so many different things. At each different place, when they're talking about hiring a data scientist, that can mean something totally different. In some places, they just want someone who can run SQL queries and numbers for every report. In other places, they want people who are actually going to build data infrastructure. In other places, they want some people who are going to build predictive statistical models and design experiments. In some places they want the unicorn that can do all that stuff. So it's really important to ask a lot of questions and figure out what a company really wants when they want a data person. What would someone actually want in that role? Are there other people currently working as data scientists at the company? What are they doing? Is there an engineering team? Is there a product team?

You mentioned earlier about working with people and making a large impact. What about the future of data science excites you the most? What are some of

the positive reasons that you would give to graduate students on why data science has greener pastures?

I'll leave out all the sociological reasons that I already talked about, why it's more enjoyable to work in a collaborative, fast paced environment, and to see the impact of your work. In academia, you don't have clients. In physics, I always felt that we had to beg people for money to do what we wanted to do, and that may still be true in data science, depending on your company. But most of the time, there are people who are interested in the output of what you're doing and really appreciate those skills.

There are lots of people who are writing tutorials explaining different techniques and different projects they've worked on. None of that existed when I was younger, and it's awesome that you can go out, get that stuff, and get an idea of what's going on.

I also think the work is exciting. It's incredibly exciting, and it's still in an early phase. I'm not going to go into the cliché of how much data we're collecting and how all of these organizations are collecting more and more data. Many people have talked more eloquently than I can about that. But it's true. Organizations have tons and tons of data, and they don't necessarily know what to do with it. They're starting to think about what to do with it, and they need help from people like us to actually do that work.

This is the reason that I came to Civis. I'm really excited about future data science applications that people are going to look at and think are benefiting society in a positive way. Like working with non-profits to use their data in smarter ways, or working with all the data that cities are releasing now.

Opening up public data is great, but there are not many cities that are using their data in a real, predictive way right now. New York has done some really interesting predictive things. Chicago has released a lot of data, but Chicago hasn't done a lot of interesting analytics with its data as a city. They just release data to the community and hope the community will do it.

I'm excited about future applications where data science is going to be seen as a positive force for good, because honestly I'm a little worried. Right now, a lot of the applications we have with data have to do with targeted advertising, cookie collection, online optimization of ad click rates, etc. That's great, but I'm worried that, at some point, there's going to be a backlash against collecting more and more data about people. I'm hoping that before that happens or when that happens, there are enough positive

counter examples of ways data is being used to benefit people and society that it can prevent some of that backlash.

I wish I could talk about this a little more specifically with the clients we're working with at Civis, because that's something we're really focused on. Before I started here, one of the big engagements we had was with the College Board, the folks who administer the SAT exam. We spent a very long time working with their data and helping them build models to understand which kids weren't going to colleges and universities commensurate with their abilities. Could we predict that? If so, what are the implications for designing interventions to help those high school students? I'm hoping that we'll have more and more examples of data science work like that, work that people feel good about rather than just seeing that companies are trying to collect data from them.

Also, one of the data scientists from the campaign started a Data Science for Social Good Fellowship in Chicago, where I was a volunteer mentor. Some of the projects we worked on addressed really interesting social impact problems, and I'm hoping there will be more and more of these kinds of applications in the future. That's what excites me about the future of data science.

When we spoke to Jace from Khan Academy, it was inspiring to see him apply his knowledge from quantitative finance to education. How can we encourage more of this in the nascent data science community?

I do worry that there's a little bit of hype, but it's undeniable that there's a very solid grain of truth to the whole data science thing.

I think people really want to do more and more of this kind of work. I think about this a lot. My wife is a lawyer, and almost all lawyers do some amount of pro bono work in a given year. I think it would be awesome if we could get some engineers and computer scientists and data people to do a certain amount of pro bono hours every year working with a non-profit. A lot of people are already doing that as a volunteer thing, but if we could institutionalize that, I think that would really be awesome for the field.

Your background as a science teacher and as a writer is different than most of the other people we've interviewed. As a science teacher and writer, how is data science doing on the PR side? What is data science missing on the teaching and writing side?

I forgot to mention that earlier. I was briefly a science journalist. I took a summer off and worked at *The Economist* and wrote about science and technology. I freelanced for

them for a while after I was in London for that summer. I actually think teaching and writing have helped me become a better data scientist because a lot of what I do is interact with my colleagues on a daily basis. I teach them new things all the time. They teach me things. We sit in meetings and look at graphs and talk through algorithms and techniques, and we ask each other questions. We explain things to each other, and you tell a story about the data. That's very similar to the things you do in a classroom when you're teaching people something. It's very similar. When you're writing about science, you try to simplify things and explain them to people. Those skills have been useful for me as a data scientist.

As a field, I think data science is doing a pretty good job. There are so many people who are blogging about their work and telling stories about their work. There are lots of people who are writing tutorials explaining different techniques and different projects they've worked on. None of that existed when I was younger, and it's awesome that you can go out, get that stuff, and get an idea of what's going on. I think that is really awesome.

Sometimes, when we talk with people who come from an academic background, they are suspicious of data science. They think of it as a fad. I'm thinking about the hype that might be behind that or how some people react to it. What would you say to somebody who thinks it's a fad?

First of all, I think it's a valid concern. I do worry that data science is being super hyped up right now. Not by the people that are doing it or who know what it's really about, but there are lots of companies who want to sell people things. A few journalists write an article on something, and everyone else feels like they need to write an article too. Then, it becomes this big giant thing.

This is why I'm excited about more positive examples of data applications. I think the more positive examples of data science that we have, the more it will help counteract a lot of the hype.

I do worry that there's a little bit of hype, but it's undeniable that there's a very solid grain of truth to the whole data science thing. We do have lots and lots of data, and we're collecting more every day. I can't imagine that companies and organizations are going to want to be less efficient in the future about how they reach out to people, about how they optimize their own operations. I think that trend is going to continue, and they're going to want people to help them analyze that data. The skills you need to do that just don't come from a single discipline like statistics or computer science. They have all the interdisciplinary aspects of what people call data science.

This is why I'm excited about more positive examples of data applications. I think the more positive examples of data science that we have, the more it will help counteract a lot of the hype. I think all the hype has not just been around data science but about tools. Everyone's been talking about Hadoop. Hadoop is great, but it's a tool. It's not the most important thing in the world, and not every organization needs to have a giant Hadoop cluster, but, with the hype, the message is, "If you're not running a Hadoop cluster, you're not doing anything interesting with your data."

The term big data makes me want to throw up because it's become an overused, overhyped thing. To me, it's not the amount of data you have. It's what you do with the data you have and how you apply it to problems and what interesting things you're doing with it. That's so much more important.

I actually don't think that anything we did on the campaign, when you talk to someone from Silicon Valley, counts as big data. We didn't have a petabyte of data, but it was what we did with it, how we were changing the organization and the practices of the campaign that was really important.

MICHAEL HOCHSTER

Director of Data Science at LinkedIn

The Importance of Developing Data Sense

Michael Hochster's path into the field of data science took a series of winding turns. After graduating from high school, Michael believed himself to be done with math. Yet, he eventually received his bachelors in pure mathematics from UC Berkeley.

After graduating and seeking something more practical, Michael entering law school, but quickly learned that it wasn't for him. After leaving law school, he eventually ended up enrolling in the Statistics PhD program at Stanford, despite having minimal exposure to statistics.

After some time spent in industry, including stints at a pharmaceutical company, a web startup, Google and Microsoft, Michael became the Director of Data Science at LinkedIn.

While we interviewed Michael when he was in his role at LinkedIn, he has since become the Director of Research at the music company, Pandora.

Your background spans both pure mathematics at the undergraduate level, law studies and then a Ph.D. in statistics. How did that happen? Can you talk a little bit about that process?

I have never been great at long-term planning for anything, so I've always followed my nose. Even the decision to go into math in the first place — I'd decided at the end of high school that I was finished with math and that I didn't really enjoy it that much. But I kept on taking just one more class in college. Then at a certain point, that was the only thing I had taken enough classes in to have a major. So I ended up majoring in math. I did end up liking it a lot, but that wasn't a directed decision.

I was mostly interested in the pure math. My dad is a mathematician so that might have something to do with it. More than anything, I really liked the fields of abstract algebra, topology, logic, and set theory. I didn't take any statistics courses as an undergraduate, although I studied probability if that counts.

Then at the end of college, my thought was, "*This is too abstract. This is fun, but I need to do something connected to the real world.*" At that time, I didn't see math as something

that was going to lead me to a connection with the real world. So I speculatively decided to go off to law school with the reasoning: "*I'll probably be able to find something useful and interesting to do with this.*"

Law seems to have a logical aspect, which is similar to math in some ways.

I'm guessing that the logic part translated from math to law very well.

Well, that's what I thought. There's a superficial similarity: it's reasoning. It's making your career out of reasoning. I always liked the reasoning part of math; it was one of the more appealing aspects. This was the story that I told myself when I went off to law school.

At the end of college, my thought was, "This is too abstract. This is fun, but I need to do something connected to the real world." At that time, I didn't see math as something that was going to lead me to a connection with the real world.

So I went to law school for a year and it was a bit of a mixed bag. I did like law school; I enjoyed the classes and liked my fellow students. But at a certain point, I started realizing that the "*I will probably find something interesting to do with this*" thinking wasn't going to pan out if I didn't have a real plan. It seemed like everybody was defaulting to corporate law. If you were taking the next step up the ladder, you went to a good law school, you went to work at a good firm, and then you became a partner.

So there was a certain point during the summer when you were supposed to find a law firm for an internship. As I was sitting in these interviews explaining to the interviewers why I wanted to work for their firm over the summer, I realized that I didn't really want to do that. I didn't really know what I did want to do and it seemed misguided. I was just going down a path that didn't make sense.

I didn't know what I was going to do next, aside from not-law-school. I did have a friend who was doing a Ph.D. in statistics at Berkeley and I got in touch with her. She told me about some of the things she was working on (she's now a professor of Biostatistics at the University of Florida). It just sounded interesting, at least more interesting than the things I was doing. Again, this was completely speculative. I didn't know anything, but I just thought, "*I know some math so I'll probably be able to pick this up.*"

So I started applying to graduate schools in statistics and took some time off. I was a temp for a while. Eventually, I got into graduate school. It was funny because all I had was a general idea that I wanted to do something connected to the real world. I wanted

to use my math background, but I didn't know anything. I didn't know what confidence interval was, I didn't know what a t-test was. I didn't know anything.

Did you find the statistics graduate school experience to be more closely aligned with what you were expecting?

Graduate school was rough for me, because of my lack of background going in. I didn't understand the point of a lot of things we were studying. In grad school, you immediately launch into theoretical statistics; you don't really have any idea about its applications. Unbiased minimum variance estimator? Who cares?

It seems like you still had the question of, "what is this applied for?"

Yes, in the end, the "*I want to do something connected to the real world*" fell by the wayside, in the sense that I was most comfortable doing math. Even though my math was a little rusty at this point, my graduate school experience resulted in my being most comfortable in the theoretical side of statistics where I could prove theorems and know I was getting the right answer. That's the gratification you get from math.

I ended up, first of all, being very far behind in understanding the point of statistics. Ultimately I fiddled around with it and tried to work out what to study. I ended up in something that is very theoretical. Which I enjoyed in a way, but it didn't have the *connected-to-the-real-world* feeling. It was very tenuous.

It wasn't until I entered the workforce that I started understanding what statistics were useful for. It was a pleasant surprise that I actually like the job, and am well suited to it. I'm more interested in working out how to do useful things with data than I am in proving theorems. It was only after school was over that I started to figure out it was a very good fit for me. It was accidental.

Even after having finished the Ph.D., I felt I was going up the next rung of the ladder. You get your Ph.D. from a good school and then you want to get a postdoc at a good school or a tenure track position, that's the next step up. On top of that, both my parents are academics, so that path seemed natural to me. And I wasn't too well-informed about other options. But in the end, it was clear to me that I wasn't passionate enough about proving theorems in theoretical statistics to make that my livelihood.

It sounds like from your story that after you finished your Ph.D., you then evaluated your options and said, "I don't really want to go down the full professor route." Is that correct?

It was a mixture of things, that's a somewhat nice revision of it. I applied for a few things,

didn't get anything. Then it became a question of, "*should I really press hard and apply and go to a place where I may not really want to live? Or should I think about what else I can do?*"

It wasn't that clear. This was a time when a lot of people were going into finance. You could go into the pharmaceutical industry. That seemed about it. It wasn't like you type "data science" into a search engine and you get fifty million jobs. I was looking for statistician or quant, something or other. I applied for some finance jobs and I ended up meandering into this position as a biostatistician for a pharmaceutical company.

Was that was the first time you were confronted with real data?

It was an interesting experience. Again, I was going in as with every step, very ignorant. I still think especially here in the Valley, people don't appreciate the huge role traditional statistics plays in the pharmaceutical industry. These companies employ armies of statisticians who are really good people. All this A/B testing that's so in vogue now has been thoroughly worked out in the pharmaceutical industry. All the philosophical hand wringing: "can you peek at the data," "Bayesian versus frequentist" and looking at subgroups. Those people have really thought about this. So that was fun for me.

Pharmaceuticals is such a heavily regulated industry. It's not just about creatively doing the best analysis you can give. It's doing the best analysis you can do within the constraints of the regulations.

I started out at Schering-Plough, a pharmaceutical company in New Jersey, where statisticians were divided into preclinical and a clinical side. I started in the preclinical group at Schering-Plough, which was statistical consulting for the scientists researching new treatments. That was the role. So it was very open-ended and some of it was interesting. This was when gene microarray technology was first getting started. They had acquired a microarray company and there was some interesting data there.

The stuff didn't work very well. The errors were huge and they needed some statisticians to work it out. They may not have been aware of this, but they needed statisticians to estimate the signal-to-noise ratio. There was a lot of explaining of t-tests to scientists as well. It was a mixture of things.

The clinical side had way more statisticians, since it was all about the design of clinical trials that would ultimately end up as submissions to the FDA. That was the more serious side of the business. The research scientists could get along without us to a large degree. However, doing analysis that ends up as an FDA submission was a pretty big deal.

Pharmaceuticals is such a heavily regulated industry. It's not just about creatively doing the best analysis you can give. It's doing the best analysis you can do within the constraints of the regulations. Some really clever Bayesian thing isn't going to fly.

The thing that was a little bit unappealing about it was it can have a bit of a lawyerish flavor. Yes, you are trying to do the best analysis. But it's not like your employers don't care which way it comes out. It's almost impossible not to look at it as finding the best statistical case.

The methodology requires that it be rigorous in some sense. But there's always a funny space of choosing what analysis to do and convincing yourself what's the best.

So with that experience in hand, were you motivated to transition to the tech industry in the late 1990s when technology was becoming big?

That's exactly what happened. I switched from Schering-Plough in 2000, which is when the web had arrived and it was the first dot-com boom. I was living in New York City and had an opportunity to work at this dot-com start up in the city. I got on board with that and I started working for a company that was basically doing customer satisfaction surveys on the web. So I ended up having an analytical role. It was a startup that had been acquired by a larger company so it wasn't exactly a startup. That's how I got started in technology.

What data questions were you asking at these web companies? What was the nature of the analysis?

The idea was to do quick, popup customer satisfaction surveys that would be lightweight enough so that people were willing to fill them out, so you could get a decent response rate. That was the idea, rather than having an extensive survey where you would have to give people a lot of incentive to do it, and then have to deal with all the biases that you incur by doing that. The goal was, you have a website that wants to know how it's doing and so it wants to ask its users some questions. This still exists today.

What was interesting was doing some analysis such as which features of a website are driving overall satisfaction on the website. That's an example of a question. The design that they had was each customer had a large set of questions and then to make the survey short, they'd take a short sample of the questions. So you'd have a list of 50 questions, but you'd only ask each user 5 of them.

Then you have some data that's massively missing, a survey that's only one-tenth complete for each user. Then you want to do some analysis with that data, where every row is mostly missing.

So it's a cool problem. There are also a few interesting methodological questions about the right way to do this sampling. One is how to make the popup random, whatever that means. There was a lot of writing SQL and trying to keep the lights on as well. It wasn't a huge fraction of methodologically sexy work but it was fun.

It's very hard to answer these questions, but the goal was to get answers to questions like, "*Which aspects of the site did customers get the most satisfaction from?*" So you've got a few problems here, one of which is that it's all correlation. For instance, customer satisfaction appears to be highly correlated with appearance, but that's because good-looking sites are also the most satisfying in other ways. So it's not that if every crap site made itself look better then it would be more satisfying.

Did any of your experience in grad school carry over to this area? Or were you learning on the fly?

Most of the time the advanced stuff I learned at school did not directly apply in any of my jobs because most of the math I learned at school was very specialized. But in the first couple of years you get some understanding of how things like regression work. Then when you come to a problem where you have missing data in every row you think, "*If I want to do linear regression that only depends on the first two moments of everything, I can estimate those fairly well. Because it's balanced in a particular way. If I just fit the regression using the first two moments of everything that might actually work, and not be too biased... And well if the covariance matrix isn't positive-definite then maybe I can fix that up.*"

In any real life work experience, you never hit anything that's just a textbook problem. It always has some weird aspect.

So there's a little bit of basic math that you learn that gives you the foundations to feel comfortable when you hit something strange. In all these situations you're always hitting problems that are slightly weird. In any real life work experience, you never hit anything that's just a textbook problem. It always has some weird aspect. The more advanced your education and the more work experience you have, the more comfortable you are about hitting something weird and figuring out how to adapt what you know.

You did metric design at several different companies including Google. Based on the different things you've seen, how do you approach the problem of metric measuring and also experimental design and knowing what to collect?

That's a pretty big question. Let's take that one piece at a time. I'll just make one point about experimental design which I think is a subtle point. Doing A/B testing is actually really hard to do well. One thing I learned at Google is you get huge value when you're

testing something if you can isolate the effect of your test as much as possible. So for example, if you're making a change to Google's ranking algorithm and you want to evaluate it, and that change only affects a small number of queries, you need to look at the queries that are going to be affected. You may not know which those are in advance, but you have to focus on those, otherwise you're not going to be able to see anything small if you pull a random sample of queries.

That sounds obvious, but it's not. Without being too specific, I'll say that it has not always been done, that approach has not always been taken. But the whole notion of isolating the treatment effect — that has a lot of ramifications that get complicated.

For example, you think it's straightforward. You just want to compare A with B. Treatment versus control, it's very straightforward. Treatment on one side, control on the other and once you figure out your success metric and you measure both sides, you're finished.

You can do that. But you might end up with a very noisy comparison if you do that — if the treatment is only targeted to a low number of samples. What you really want to do is compare those subjects that were exposed to the treatment, that were affected, to those in the control group who would have been affected by the treatment.

So this counterfactual comparison is what you are really interested in. Because it means that if you're doing things the right way, you need to be logging on the control side whether each subject would have been exposed to the treatment. This means that you can't just use your standard production, unless your standard production involves logging counterfactually what treatments would have accomplished.

It's often not so obvious how to identify, in the control, whether they would have received the treatment. So I've seen that people usually ignore this. That's what I would have done starting out, it's only because I've worked through these problems at Google that I realized we're not seeing anything because we're comparing the whole treatment to the whole control. It's just swamped in noise.

This has been a running theme for me, trying to hammer home this point. There are a lot of cases, even at Google (where experimentation is an extremely well-oiled machine), where this kind of thinking is not carried as far as it could be.

How did you balance the theoretical rigor you were coming in with, given your academic training, with the practical demands of actually applying statistics in an industry setting? Were there tradeoffs you had to make?

That's definitely a constant challenge.

There's a more basic thing, especially where there's a division between engineering and analysis — then who decides what gets logged? From what I say it's clear it matters a lot what's logged. You may need to log a lot of things to do it correctly and it might not be obvious that the company needs to log it.

Thus, even if you want to do the basics, you ask engineering to do some work that's not very interesting and a pain. There's a reaction of, "*Really? I have to log that? Explain to me why I need to do this.*"

The more nuanced the reason, the harder it is to do. It's hard to demonstrate the value of it until it's already in place and then you can say, "*Look at this analysis that we were able to do.*" It's 100x more informative than it would have been if you hadn't done this extra logging. So yes, it's a challenge.

The closer and better-aligned engineering and analysis is, the less friction you get.

It seems that analysis spans many different domains of the company. Not only do you have to work with engineering, but also once you update your analysis then you need to show it to someone who will act on it.

I do think the communication needs of the data scientist role is one of the most important things. When I'm hiring, there are always some trade-offs between different skills, but the ability to communicate well is a given. Because it's important in so many ways, both in negotiations with other teams and making your analysis have an impact on the organization. You have to be able to talk to people and explain why it matters.

The subtext of "I'm smart" doesn't matter anywhere outside of graduate school. You have to start with: Here's what I found and why you should care about it.

There are people who are good at analysis and people who are good at writing code. For some of these people, there's such a strong temptation to present things as, "*I did this, and then I did this, and then I did this. And then I did this really smart thing and here are the results after I did all these really smart things.*"

No one gives a shit about that. No one really cares how smart you are. This is why it's different from graduate school. The subtext of "*I'm smart*" doesn't matter anywhere outside of graduate school. You have to start with: *Here's what I found and why you should care about it.*

Seeing as you've worked intensely with understanding metrics at numerous

companies over the past few years, what have been some of your important takeaways?

I have two main things to say about metrics.

First, there are two very different angles of looking at metrics. There's the overall evaluation criterion idea, where you think of a metric that everybody agrees represents progress. Then you focus all your efforts on improving that metric with the understanding that this is our understanding of when progress is made.

So when I worked at Microsoft, this philosophy was very strongly advocated. "This is our overall evaluation criterion, you have to move this. I'm sorry but if you can't move this then it isn't worth shipping. Too bad."

My philosophy of metrics is almost the opposite of this. I think overall the evaluation criterion is good — you want something you can track. You want to have a number you can believe in that represents the overall health of your product. But generally, anything that's accepted as the driving metric in this philosophy is going to be too broad and you're not going to be able to move it. So you really have to make a conceptual distinction in my opinion between approximate metrics — that you use to decide whether a feature is good, and which are going to be very fine-grained and specific to your feature — and these overall global metrics that you hope go up.

It depends a little bit if you're in a business that depends on small improvements or whether you're at a stage where you're making many big improvements. For example, in Google search many of the improvements were small. If you're improving the ranking, it's small. If you restrict yourself to these broad metrics you can never ship anything because they don't register on the global metrics.

One example of a broad metric is searches per unique user. Using that you can do something to ranking and your search engine gets better so people might use it more. But that's really hard; you're not going to be able to see that in most experiments.

So I am very big on thinking very hard about what a particular feature is trying to accomplish, and how we can measure that as narrowly as possible. It's good to have a small number of metrics, but for shipping you want something that measures as closely as possible the positive impact you expect that feature to have, and not think of it as an overall evaluation criterion. So my philosophy is the opposite of what I saw at Microsoft.

The second thing about metrics is that you don't have metrics on the metrics. You never really know what a good metric is. So you spend a lot of intellectual cycles trying to

develop things that are useful. And these things are being used as a yardstick for other things. So you never have clear guidance about whether your metrics themselves are any good.

It comes back to getting a new metric established. How do you do that? You have to lawyer it again. That's tough. How do you do that? How do you go about convincing yourself that a metric is good? And how do you go about selling it or convincing other people that it's good?

It's really hard. A lot of the time, at least in the domains I've worked in, you're interested in "is a particular feature good?" Or, "Is a particular change to a website good?" You don't have access to seeing the electrodes in users' brains to know if it's good or not, so you end up inferring what's a positive impact from behavioral data. While there are all sorts of things you can do to try and quantify user behavior, the issue is that you never really have absolute truth about what's good.

So one possibility is to try a lot of things that seem plausible. You don't have absolute truth but you have correlation: lots of plausible approaches that seem to point in the same direction.

It's not logically guaranteed, but if you start with some plausible things, a lot of them move together. Some seem much cleaner than the others. That's how you proceed. It could be something else that's making all these things move together, but it's also plausible that there's underlying goodness that's driving them all. So I start to believe that.

The other way, that's somewhat of an empirical approach. You observe different possible things and the fact that they've all moved together gives you some interactive evidence that they're all doing something reasonable.

This is a really good point. Could tell a brief story about one of these instances? I understand the lesson in the general case, but, what was one thing that you tried to measure that was hard to get at, and what were some things that you designed to tackle this problem?

I'm going to go back to the example of search.

Let's say you want to use clicks on a search as a measure of accuracy of search results. Well, a lot of that is based on a "more-clicks-is-better" principle — you start thinking about when more clicks are better. If I have a query, "What is the capital of Albania" and I have a lot of clicks on that query, is that a good thing? Probably not. But if I have a

query such as "Best digital camera" and I have a lot of clicks for that kind of query, that probably is a good thing.

First you take a proposal for a metric. Even without looking at any data, you start thinking about under what circumstances the metric could point the opposite way that it's supposed to. Then you get some ideas; you see there are some major cases where the metric points the wrong way. Maybe we need some segmentation. Maybe there is no simple metric that is going to cover everything; maybe we have to consider navigational queries and more browsing queries separately to have any behavioral metric that makes sense.

You can convince yourself by introspection. The story also becomes more convincing when you observe how this new segmented metric compares in practice to just counting clicks. Neither of those things is entirely satisfying. Because your empirical stuff isn't conclusive, your empirical evidence is never conclusive. Your thought experiments are just thought experiments. So it's not math. It's not science.

This sounds like a deep distinction between academia and industry. How has industry changed your view of math?

I still like doing math puzzles for fun. I still think math is beautiful. But in terms of when I think about math as something on the job for me, I don't consider myself any kind of mathematician. It's just there, it's a tool.

I'll say something a little broader than that: math is just there as something that will help me figure out my problem. I feel the same way about all the machinery of data science.

For me it's the actual substantive questions. For me, there's data, there are interesting questions. I want to answer the questions. There are interesting products you can make that require the data to be big and that's cool. But the big data *per se* isn't interesting. Maybe I've been spoiled a little bit at Google where you have this massive infrastructure and you don't need to think about it that much. You just write a script and send it off and Google's massive infrastructure processes it, and you get your answer back.

Let's talk about your view on data science. How do you see the term, and the division within the roles in the field?

It's a good question. I think I will probably say something similar to what Pete Skomoroch says, although I'm pretty much all the way type A. For me, a type A data scientist is about Analysis. B is for Building. Of course there's no dividing line between those things. A lot of people do some of both.

At LinkedIn, it's divided up that way. There's a team called Decision Sciences. They are focused more, but not exclusively, on analysis, working with product teams, experimentation, some model building. But for the most part it's far from putting stuff into production.

And then there's the data products team which uses data science techniques to build things. So I think that's a reasonable distinction to make.

So why even have this term "data science"? What's new here? Why don't we just call type the A data science a "Statistician" and type B something else? Well, I do think there's a little bit more to say than just that.

There's a lot of practical work with data that is not covered, and it's totally different from the statistics curriculum. I'm talking about all the practicalities of data, all the visualization, the critical aspect of it and the communication piece that we talked about earlier.

Statistics as a field has focused itself very narrowly as a field on producing certain artifacts: confidence intervals, hypothesis tests, p-values etc. These are the work products of the statistician. In the pharmaceutical industry, that's what it is. You have a report but at the end of the day you're saying, "Here's my p-value and it's less than .05."

However, for data science, there is so much more to it than that. Although I have to admit the term "data scientist" took me a while to get used to. I was a late adopter. I laughed when my friend sent me this meme that said: "*Data scientist, is that like a hammer carpenter?*" I thought that was funny. But I accept it now as a term that encompasses much more than statistics.

I laughed when my friend sent me this meme that said: "Data scientist, is that like a hammer carpenter?" I thought that was funny. But I accept it now as a term that encompasses much more than statistics.

I think that the type B data scientist — the data engineer — I think that's a reasonable distinction to make. It really amounts to: these people all have some mix of statistics and analysis skills and coding. And then you can make a continuum or you can say you're mostly stats or you're mostly engineering. Josh Wills defines a data scientist as someone who's a better coder than every statistician and a better statistician than every coder.

To me there's probably no-one in that category. At Google, the really strong coders who know a lot of statistics are software engineers. They don't have a special title. They're just software engineers who know a lot about machine learning. They might call themselves

a data scientist to get a job. I can see that point of view.

Sometimes you get people who are quite good coders, but they're not all the way to being a software engineer. But they know a lot of machine learning. So they can implement prototypes, but not all the way to production. That I think becomes a tricky place for data scientists because you're in between things. But I still consider it to be a useful designation.

With that designation in mind, when you are trying to hire people who are coming out of school right now, what are the features you're looking for? And then more generally, how do you think when you're building a data science team? What goes into that composition?

I'm not sure I can answer the second question that well, since it depends on what you're trying to do. At LinkedIn, I'm not exactly the person who built the team. I came into a team that was already there. When I think about hiring, I want people who can code somewhat, although I myself am not a particularly good coder. But we're more focused on analysis. So when I'm looking for people, the most important qualities that a data scientist should have include having a feeling of how to take a data set and answer a question with it. Figuring out what should I compare? What's the control? What's the way of transforming the things that I have available to me to make it reasonable? What am I missing here that I need to go and collect?

This isn't stuff you learn in school. Some people have it; it comes from experience too. It definitely comes from working with data. So I look for people who have real experience with data – whether it's in a hard science, a social science, computer science, or statistics. Just understanding theory isn't enough. You need data sense.

Just understanding theory isn't enough. You need data sense.

I'm also really looking for the ability to communicate well about things you've done, and good judgment. Understanding that when you're working through a problem you have a series of choices to make and being very aware of the choices you're making and why you're making them at every stage. That's part of data sense too. So these are somewhat intangible factors.

I also look for some facility with coding — you need to be able to get your data and manipulate it. Therefore coding is required. I myself don't look for super-heavy coding skills because I feel like a lot things in my world have to be picked up. Also, I can't evaluate it myself when I talk to people.

Then, the more formal statistical inference is the last thing that I look for. Not that it's unimportant, but it is probably last.

From what you've said, it seems that what's most important is to have real life experience working with data.

Not everyone who has worked with data gets the data sense that I'm trying to pick out. But working with real data seems to help with that. It's one of the factors.

I spent so many years at Google inventing and asking people math brainteaser questions, which is basically a "*How smart are you?*" type of gauge. I've totally given up on that stuff, because although it measures something — people who can do all those things, they tend to be smart people — there's a job opening that I have and I don't think an ability to answer these questions is that strongly correlated with being able to do the job well.

KUNAL PUNERA Co-Founder/CTO at Bento Labs

Data Mining, Data Products, and Entrepreneurship

Kunal Punera started hacking on computers at an early age in India. Inspired by the way Google transformed internet search through indexing and information retrieval, he came to the United States to do his PhD in data mining and machine learning at UT Austin.

After for years at Yahoo Research working on diverse data problems, he joined Customer Relationship Management (CRM) startup RelateIQ, as their fourth engineer and first data scientist. At RelateIQ, Kunal built the data mining system from scratch — as well as many of the data products deployed.

Recently, Kunal recently left RelateIQ to start his own company, Bento Labs. RelateIQ was acquired by Salesforce for $380M. In this interview, Kunal shares his experiences bridging from research to data science, thoughtful lessons about data science engineering, and the importance of tool making.

Let's start with where you are and where you came from. Starting from undergrad, what was your journey? Why did you go into data?

I did my undergraduate work in computer science in India. During that period I was more of a hacker, building things without being too worried about the theoretical side. At some point, towards the end of my undergraduate studies, I received all these offers to work for some software companies and to just hack on things. I didn't quite know much about the Master's or Ph.D. programs in other parts of the world. Then, one of my close friends was accepted into a Ph.D. program in the U.S., and I started hearing the vocabulary surrounding graduate schools in the U.S., GRE scores, the application process, etc. That's when I started considering the possibility of studying further.

I was not really sure at first if I wanted to pursue further studies. So after finishing my undergraduate studies, I took a year off from the software companies to work with a professor and helping with his research; if I was going to commit many years of my life to research, I wanted to first see if I would enjoy the work. And what I discovered was that I loved it. I loved research just as much as I loved ad hoc hacking. The professor, Dr. Soumen Chakrabarti, and I wrote a couple of papers on data mining that ended up being published at the 2002 World Wide Web conference.

The way I got started on data mining was coincidental. From my hacking experience, I was interested in databases and operating systems, but Soumen told me that OS and DB research was already pretty mature, and that there was a new field of research that involved combining artificial intelligence and data that he needed help with, so that is how I started working on data mining problems. Some of the first projects I worked on were on web mining and machine learning for the Web. I enjoyed thinking about those problems and really got into them. There was a sense that I could have a tangible effect on the lives of users through my research, which was pretty exciting.

To provide some context, this was back in 2001 and I had just discovered Google. It provided a real life example of "what data mining can accomplish." The other search engine at the time, AltaVista, was not nearly as strong as Google. You could clearly see the difference in search quality. This was one of the first times that I saw how data mining could make a huge difference in the way people live their day-to-day lives, how they accessed information, and how they behaved online.

This was back in 2001 and I had just discovered Google. It provided a real life example of "what data mining can accomplish....This was one of the first times that I saw how data mining could make a huge difference in the way people live their day-to-day lives, how they accessed information, and how they behaved online."

After working with Soumen for a year, I applied and was accepted into graduate school at the University of Texas (UT-Austin). My Ph.D. advisor there, Dr. Joydeep Ghosh, gave me a lot of space to explore various problems in data mining and machine learning. It took me a little while to get into the academic mindset – I spent my first 2-3 years exploring the new country, with road trips across the U.S., and weeks spent in national parks. I also spent a lot of time at internships at industrial labs – IBM Almaden and Yahoo! Research. I finally got serious about research in my third year as a graduate student, and did some good work in my fourth and fifth years to wrap up my Ph.D.

What was your Ph.D. in?

Topics in machine learning and data mining. The topic specifically was classification in the presence of structure in data. If you have structure in the data, could you use that to make learning algorithms better by enforcing constraints? I was very motivated by real world problems. My last two years of Ph.D. were funded by Yahoo! so the problems I tackled tended to be rooted in challenges Yahoo! was facing at the time: searching and indexing, classifying web pages, and trying to model user preferences and behavior. I solved a bunch of real world problems, and the thing I found in common between the

different solutions was that they always exploited the structure in the data – websites are hierarchical structures, web pages as well, and people's browsing could be modeled as Directed Acyclic Graphs (DAGs).

In reflection, my Ph.D. thesis topic came about more from figuring out a common thread in the different problems I worked on, as opposed to having a particular research agenda and pursuing it. That's been my approach to research since my Ph.D. as well. I don't have a specific agenda or an area that I'd like to advance. I don't really feel like pushing any one particular technology. All I want to do is solve hard and interesting problems. After my Ph.D., I took a full-time position at Yahoo! Research which, in those days, was almost an academic organization. It was basically a university department without the teaching load. It was perfect because I basically went from being a graduate student to a similar place, but was paid significantly more. Plus, they had the benefit of having an immense amount of data as well as great infrastructure to work with.

What kind of problems were you solving there? Were they motivated by Yahoo! user-facing business?

I don't have a specific agenda or an area that I'd like to advance. I don't really feel like pushing any one particular technology. All I want to do is solve hard and interesting problems.

At Yahoo! Research, we had a pretty open charter to help direct our work. 50% of our time was to be spent making an impact for Yahoo! and the remaining half could be spent working on research problems that may not have much to do with the immediate needs of the company. That was an amazing experience because I got to touch pretty much any problem I wanted to. Since I could code as well as do research, I got to work on a lot of different data mining problems – including designing better CAPTCHAs, email spam detection, phishing detection, search engine ranking, targeting for the advertising systems, and new approaches to user modeling. I spent four years at Yahoo! Research and worked on a wide variety of projects ranging from short-term ones (3-6 months) to some engagements that lasted years.

Did you focus mostly on the research aspect of coding, or did you also deploy your research, such as spam filters, in production?

Research scientists at Yahoo! were judged based not only on the amount of internal impact that we had, but also on the number of research papers we wrote, the number of external talks we gave, etc. Typically, Yahoo! Research did not own the projects I was working on, and so I had to work closely with the products teams responsible for them. Given the nature of the engagement, naturally, the product teams were hesitant to have

researchers make direct changes to the code base; the products teams had a focused agenda while I had a broader agenda and different responsibilities. But I had a strong development background and wanted to build/optimize the systems end-to-end, and therefore, I felt a little frustrated.

Also, after years in research, I realized that academic careers require one to be an expert in a specific, narrow area. There's a lot of pressure to become an expert at one thing, so everyone would know Kunal Punera is an expert at so-and-so. My research agenda has always been "show me an interesting problem and I will work towards solving it." Over my four years at Yahoo! I moved across a lot of different types of problems and domains, ranking problems in search and ads to adversarial data mining in mail spam and CAPTCHAs. That doesn't mesh very well with how academia expects publications and careers to progress. Eventually, I realized that a purely academic career wouldn't sustain my interest enough. Moreover, outside Yahoo! I was watching the emergence of these interesting companies that are using data to solve important problems for people, and I really wanted to get involved with that.

Moreover, outside Yahoo! I was watching the emergence of these interesting companies that are using data to solve important problems for people, and I really wanted to get involved with that.

At some point, I started considering leaving Yahoo!, and maybe starting my own company. During this process I realized that I had been away from software development for too long. During the five years of my Ph.D. work and four years at Yahoo! Research, I had written a lot of code, but code written for research doesn't have to be production-quality; it doesn't have to be maintainable, it isn't typically changed by anyone else. Also, the whole world of web development had undergone considerable change since the time I had been writing systems in cgi-perl. You probably don't even know what that is — it was the precursor to the modem web application frameworks. I realized that I had to update my knowledge about software development, especially when it came to using the open source stack.

So I realized that I had much to learn before I could start my own company, and that I wouldn't be able to do it while at Yahoo! I had to go work at a place that would appreciate the fact that I had a very strong research background, but would give me the opportunity to learn stuff beyond data mining, and RelateIQ fell right in the sweet spot. The match was amazing. The founders were interested in building a company that solved key pain points around relationship management using cutting-edge data mining. Furthermore, since I was to be the fourth engineer, I would have to build everything from scratch on my own, and consequently, would learn a lot from the experience.

I spent two years at RelateIQ. I worked on building the data mining system from scratch — and by the time I left I had built most of the data products deployed in RelateIQ. And in the process I learnt a hell of a lot.

Wow! How did you learn all of this on the job, and on the fly?

I spent two years at RelateIQ. I worked on building the data mining system from scratch — and by the time I left I had built most of the data products deployed in RelateIQ. And in the process I learnt a hell of a lot.

My data mining background was very deep and broad, so I didn't really have to learn any of the learning algorithms or approaches on the fly. I picked up Natural Language Processing (NLP) as needed, but once you have a decent statistical modeling background, the rest of machine learning is just variations of the same thing. Those were not an issue. But, software development skills were something I had to learn a lot about. For example, while I was a good coder, it is a different experience to work with engineers who had worked in production environments their entire careers. So, while Maven (for dependency management) was obvious to them, it was new to me. Using Guice for dependency injection was normal to them, while it was something I was picking up for the first time.

During my time at RelateIQ, in terms of software engineering, I learnt a lot. Sometimes I feel I didn't learn nearly enough. *[Laughs.]* I think there's a lot more I could have worked on, but whatever I learnt, I learnt well. Whatever I know about machine learning algorithms, I learnt at Yahoo! Research. Whatever I know about software engineering, I learnt at RelateIQ.

I had come to RelateIQ as a stepping stone to starting something on my own. But as two years passed, RelateIQ blew up, in the sense that it was doing extremely well. So, there was a strong temptation to stay because the stock options were going to be worth a lot at some point. But then I also had some confidence that I could make something valuable of my own in the next few years. I loved the people at RelateIQ, but with a heavy heart, I made a decision to leave. If I hadn't left now I might never have had a chance to do my own startup.

I left RelateIQ to do my own startup. In the last two months, I've just been rebuilding many of the things that had always existed at RelateIQ. I've been building a backend and figuring out how to get continuous deployment working, learning how to get the database to perform well — all these things which I didn't have to do before. It's been interesting.

That's amazing. It seems like you've systematically identified areas of knowledge you wanted and found either employment or career opportunities where you could go and be paid to learn those things. Once you mastered those things, you can go to other things. How did you do that?

Silicon Valley is the place where, first of all, there's a huge demand for the kinds of skills we have built up. I'm lucky that I've chosen to work on something that has a huge demand, and companies are willing to let you learn on the job as long as you can contribute back. RelateIQ, for the entire time I was there, did not have another data scientist, so I had to carry that whole load, but in return, I learnt a lot. Working in startups is always a give and take, and I think good companies in Silicon Valley understand that the employees they are trying to hire are, in general, smart, and have their own long-term goals that they want to pursue. As long as they contribute a lot back to the company — and I like to think that I did — then the companies help further the goals of the employees.

Once you have a decent statistical modeling background, the rest of machine learning is just variations of the same thing. ... But, software development skills were something I had to learn a lot about. For example, while I was a good coder, it is a different experience to work with engineers who had worked in production environments their entire careers.

When I wanted to leave RelateIQ, everyone, from Adam to Steve to DJ, was very supportive. Steve wanted to introduce me to investors. DJ wanted me to come by and get his advice on the new idea. The environment was extremely supportive. We are very lucky to live in a business environment where companies don't even think about locking employees in. There's no notion of an employee lock-in. There's no notion of company lock-in. There's always flexibility. Everyone wants to make the best possible use of their time. We all understand that we're on Earth for a short time, and we all want to go to a company or work on things which uniquely need us. Silicon Valley is unique, and amazing in that way. We're lucky to be in this situation.

Being able to learn new things really quickly is one of the things we need today more than ever, but there's an art to doing that. You need to have some sort of foundation, core programming skills, core modeling skills. If you were to decompose those down to the principle skills, what do you feel is most important?

In terms of programming skills, I'm not sure what the curriculum nowadays looks like, but in my undergraduate days, I started by learning C. Actually, I learnt Pascal first. Then, I learnt C. These are pretty low-level languages with few rules and close interaction with the machine. So, I learnt from the very beginning how programming languages manage memory, what pointers are, what an execution stack looks like, etc. I think that

experience was useful because now, if I have to learn new concepts, it's easy for me to go back and reconstruct them from the first principles in my head.

Whatever I know about machine learning algorithms, I learnt at Yahoo! Research. Whatever I know about software engineering, I learnt at RelateIQ.

In terms of programming, I would think learning about core programming concepts is important. You should start learning from there. I have a feeling that if you started learning programming with Javascript, it might be a little bit more difficult to know exactly what's happening in the background. I would encourage one to learn what is happening at a low level, but also to not spend too much time on it. I spent way too many years working with C and C++. Nowadays, I wouldn't build systems in those languages. Java, Scala, Ruby, Python have amazing framework support, open source libraries, and lots of solutions documented in sources like Stack Overflow.

In terms of data modeling, I think I was lucky that I took some good statistics courses. It's useful to understand the underlying concepts of algorithms. I think a graduate-level optimization course is important, as well.

One of the obstacles I sometimes see engineers running into is confusing the core problem that needs to be solved and the one particular solution to that problem. Sometimes people have one way of solving the problem already in their head, and they might not see that the core problem is not the same thing as their solution. As much as possible, I would encourage people to constantly ask the question "What am I optimizing?" For example, if you want to obtain a clustering of data, it's useful to first try to determine what properties you would want in a good solution, and then attempt to encode these criteria into a loss function. If one is not careful it is easy to think of clustering data in terms of steps the algorithm should take, or a series of methods that must be implemented. This can sometimes lead the engineer astray in that the preconceived solution might never end up obtaining clusters with the desired properties. Of course, the time constraints in a startup don't leave data scientists the luxury of carefully thinking of every problem. In these situations, experience helps.

Do you have any specific examples for that? I know what you're saying in the abstract but it would be helpful to hear a concrete example.

For example, suppose you want to do classification. Say I have two classes, and I wanted to learn a model that separates them. I can use one of the many algorithms out there: decision trees, support vector machines (SVM), random forests, etc. But one might come to erroneously think that the classification problem is equal to learning decision trees — without completely understanding what underlying problem is being solved.

Before jumping into implementing a solution, one might want to consider some questions about the nature of the problem. The core problem here is that there is a boundary, a separation between the two classes, that we need to find. Well, what does it mean to find a boundary? What kind of boundaries do decision trees find? And what kind of boundaries do linear SVMs find? Will using a kernel method help? Is this a situation where I need to worry about irrelevant features? Does that mean I need to regularize via a L_1 norm or stick with L_2? These are fundamental questions that once answered can guide us to the appropriate approach and thus avoid a lot of trial and error. Moreover, they help in the following situation: once we have applied the first algorithm and it obtains 65% classification accuracy, what should we do next to improve the results? Carefully defining the parameters of the problems and the characteristics of a good solution help us figure out what the next step to take is.

Good companies in Silicon Valley understand that the employees they are trying to hire are, in general, smart, and have their own long-term goals that they want to pursue. As long as they contribute a lot back to the company — and I like to think that I did — then the companies help further the goals of the employees.

Sometimes when reading Hacker News, I get a sense that people feel machine learning is simply about taking open-source libraries and applying them to data. In many cases, this works okay as the first step, but, often, the next step to further improve the model is difficult to figure out. But if one has a strong understanding of what these libraries are trying to optimize for, what each core algorithm is good for, how the curse of dimensionality effects learnability, what the difference is between L1 norm and L2 norm, and other such kinds of things, then it becomes much easier to figure out how best to apply these open source resources.

So is this the set of skills you are looking for when you are interviewing for the data scientist position?

When I am looking for data scientists, the most important thing I am looking for is whether their approach to machine learning is systematic. Sometimes I meet people who know what first step to take, and can do it pretty fast because they're amazing coders, but the second step becomes a little harder. When I interview people, I don't really want them to solve anything on the whiteboard, I don't want them to code. The key thing I want to know is whether they get the underlying principles of what they are building. I'll typically ask them about something they've previously built and then just delve deeper and deeper into that same problem. I find this is a good way to evaluate candidates because if they contributed significantly to the work and know their fundamentals, they would be able to defend their decisions from first principles and not just say, "Everyone

likes SVMs so I used it." They should say, "Well, the problem had the following properties, that's why we needed a SVM". Or "I also tried this other thing. It didn't work well because I believe… " as opposed to "I just didn't try it."

So, I learnt from the very beginning how programming languages manage memory, what pointers are, what an execution stack looks like, etc. I think that experience was useful because now, if I have to learn new concepts, it's easy for me to go back and reconstruct them from the first principles in my head.

I think that's a key thing to look for — strong fundamentals. If someone has strong fundamentals, but doesn't know what a random forest is, I don't really care, because individual machine learning approaches can be easily learnt. Having a strong background and then picking up random forests is way easier than having just shallow knowledge of random forests and then trying to debug them. People who are looking to work with data mining algorithms should take a systematic approach to learning them. I think it's difficult to do that nowadays because there's so much demand for the skill set. But I would urge them to get more fundamental skills via, maybe, a machine learning course which focuses less on the specific algorithms and more on the fundamentals, and then some core statistics, optimization, and algorithms courses. These will give them a good foundation for their work.

When you were trying to build the data science team at RelateIQ, how big was the team?

At RelateIQ we were not building a *data science* team. One of the things I'm not fond of is the kind of process I used to follow at Yahoo! Research, where the science team builds the models and then passes them off to the engineers to implement or deploy. I feel like a lot gets lost in the translation. Sometimes the models that one builds assume an environment that doesn't really exist in production, and one doesn't know that until the models are deployed. And in fact, when models get deployed and accuracy lags, it's very difficult for engineers who didn't build the models to convey back what's exactly happening.

I prefer that the scientists be closely involved in the implementation of the feature pipelines and the models in production. They should know how the model is deployed, everything that happens to the data — filters, sampling — before it shows up as input into the model. If there's a particular filter which takes out one particular type of data, the scientists should know about it. Moreover, in the first few months after a model is deployed, the scientists should be the ones maintaining it.

At RelateIQ, we were following this principle and building a *data products engineering* team. We didn't call it data science at all. In the data products engineering team, we looked for people who had a very strong sense of data, who liked playing with data, but also had reasonable engineering skills so that they could actually touch production code directly. We didn't expect them to roll out their own hadoop infrastructure, though many of our data product engineering people did. But we wanted them to be able to deploy their own models, run them, write the feature extractor pipelines, etc. Apart from the principle I mentioned earlier, a second reason for building the team this way was more pragmatic; we were a small team and we couldn't spare engineers dedicated to taking the work done by data scientists to production.

It seems like you not only focused on people who can do data analysis but also on those who have a strong engineering background, people who started out hacking or coding and then went to data science.

At RelateIQ, we were following this principle and building a data products engineering team. We didn't call it data science at all. In the data products engineering team, we looked for people who had a very strong sense of data, who liked playing with data, but also had reasonable engineering skills so that they could actually touch production code directly.

I find that you can go both ways: start from the data side or software development side of things. But in a startup, having people with both skill sets is critical. One thing that you never want in a startup is to have a data scientist working alone with no established process to get work into production. I've seen many startups where data scientists are six months ahead of production systems; they have done six months of work that hasn't been deployed because the engineers that touch production are busy with their own work or fires. These data scientists have done the work in R or matlab and are not able to integrate it with the production backend system. You don't want to have that situation.

In a slightly bigger startup, one may want to have small teams of two or three people — one data scientist, one person with engineering system type skills, and one with product management type skills — to build and maintain data products. They work as a team to build features as opposed to a data science person building a model alone, and hoping that one day some engineer is going to step forward and bring those features to production.

When RelateIQ was small we avoided this situation by having one person perform all three roles — data scientist, engineer, and product management. Now that we are larger we are building multi-skilled teams.

This is a fantastic example of data science and product done right. You avoided some of the pitfalls of great, locked up models that you can't deploy. Were there other things you saw at RelateIQ that you felt were really good lessons in building data products?

One thing that you never want in a startup is to have a data scientist working alone with no established process to get work into production.

Other than the constitution of the team, another important aspect to keep in mind is the cadence of development of data products. Engineering the systems that involve data mining is a little different in the sense that most times it is not clear how many engineering resources will be needed before the models reach production quality, or even whether the desired quality can even be reached. This may not be a big problem at large companies, but it makes scheduling and resourcing data products tasks problematic when resources are constrained, as in a startup. In a startup, one should want to break down data products work such that visible, measurable progress can be made in two-three weeks so that the engineers have intermediate wins. This also prevents engineers from going too far down the wrong path. Of course, the development cycles need to be long enough so that hard problems can be attempted and solved; very short inflexible cycles typically lead to data products that have been patched together and are not robust.

Another important aspect relates to scheduling; if you want a data product deployed at any point in time, you probably should give the data engineers a head start so that they definitely have their models or features built before the front-end or back-end resources become available. This is simply a consequence of the uncertainty around the pace of progress on data products.

For a long time after I joined RelateIQ, I was the only one working on data products. In these early days, scheduling was not that much of an issue since I was the only data, backend, and frontend resource. Moreover, I have a lot of experience with data mining and was able to avoid going down bad paths, and was able to get most models deployed in the first couple of iterations. As the team grew and I had more frontend and backend resources that could help me, we had to work harder on scheduling and we applied the principles I outlined earlier.

Any other data product hacks from RelateIQ?

The other thing that we did a lot is we took shortcuts. We took shortcuts all over the place. At the beginning we were in a hurry and we didn't even know whether the products would be received well by the users, and so we tried to get away with putting in as many

hacks as possible. We made our decisions on the hacks in this particular way:

Anything that was fundamental, core-level, functionality, such the parsing of emails, we made sure it was extremely strong. There are many reasons why functionality gets put into this category: in the case of email parsing, first every email has to be parsed and the cost to reparse is very high (I'll have to go back and fetch every email and reparse it), and second, a whole bunch of other features of the data system depend on accurate parsing of email. Therefore, my system for parsing is very strong. It involves nice, sound models based on CRFs and SVMs that we learnt over large quantities of training data and that are continuously trained as data changes; these models are sound.

The other thing that we did a lot is we took shortcuts. We took shortcuts all over the place. At the beginning we were in a hurry and we didn't even know whether the products would be received well by the users, and so we tried to get away with putting in as many hacks as possible.

Other functionalities are higher level, such as automatically making a suggestion to follow-up with a contact. There are many questions that need to be answered here that are difficult to optimize using data since the degrees of freedom are too high or training data is too noisy. When a suggestion has to be created, the system has to determine if a follow-up to an email is warranted, whether the user has already done the follow-up, how long the system should wait before reminding the user to follow-up, and if multiple users are referenced in the email that the suggestion to follow-up will be directed to. The dimensionality of this space of choices is so high that the first attempt to model this should involve using manual rules and hard thresholds.

Another example is trying to learn the effectiveness of our rules for follow-up suggestions via usage data, and using the feedback if a user rejects the suggestion. Even this is complicated since rejection is a very aggregate action and it's not clear what the user is rejecting; maybe we made a mistake in parsing the email and no suggestion was warranted or maybe the user liked the suggestion but we made it too early, or the user doesn't like the sender of the email, or even the user hates all suggestions in general. So even here I shied away from modeling the entire problem and put in a lot of rules, a lot of very simple models. The first cut solution involved counting the number of rejections and rules for actions to be taken when certain thresholds are met. As the data product improved we used more advanced approaches to model user feedback.

Yet another important aspect is tools, and this is something that was driven home to me at RelateIQ. Before working at RelateIQ, I had a very high threshold for tooling. Sometimes I ended up doing the same thing 10 times without automating it because every time I did

it, I was not sure I would be doing it again. After RelateIQ I can safely say that if someone does something 3 or 4 times, they will be doing it again in the future, so they should try to automate it; automate data cleaning scripts, automate model deployments, write tools for re-training of models, don't do it by hand. Write tools that will automatically create a fresh data set, retrain the model, check its accuracy, and send you an email if the accuracy is below a threshold, and, if the accuracy is good then deploy the model. This tooling might seem excessive but is going to easily pay for itself in terms of saved time in the long run.

Was that a change of pace compared to when you were working in research?

Yet another important aspect is tools, and this is something that was driven home to me at RelateIQ.

Well, I did a lot of work at Yahoo! Research. I worked on new projects every 3 to 6 months. I was writing three to four papers every year and that's a lot of work. I'm used to working a lot of late hours. At RelateIQ what changed was the emphasis. At Yahoo! Research, the emphasis was always on doing something innovative. It was all about asking, "*Last year Microsoft Research published this. The year before, Google did. What's the new angle I can find and solve the problem?*" Sometimes the problem being considered was not an immediate concern for Yahoo. Other times the problem could be solved using simpler means. At these points, we would consider more complex versions of the problem with additional hurdles, and then figure out how to solve them. The goal was to constantly push the envelope of what was possible with data mining, and not just to solve immediate practical problems. The task was as much to find new problems as to solve them.

At RelateIQ, I worked extremely hard as well. There, the problem to be solved was pretty clear, and main question facing me was "*What is the minimum effort that I can put forth and get something into the hands of the users so that I can test whether the solution is useful?*" And from that feedback I can figure out how much more effort I want to put into it in the future to improve the feature. Moreover, the solutions I tried out were chosen not just for their innovativeness, but using a tradeoff between effectiveness and the cost of implementation and future maintenance.

So the main difference from the earlier days of research was not one of pace of work. It was a change in priorities.

How do you measure cost? Development effort or time?

Cost in this case involves implementation and future maintenance. In a startup you have

to constantly trade that off for accuracy of models. If one option is a complex model such as a Conditional Random Field (CRF), but we can come up within 5% of the accuracy using a Naïve Bayes model, then we would choose to go with the Naïve Bayes. It is not just that CRF models would be significantly harder to train within a typical project timeline in a startup, but as the data environment changes in the future, the CRF will break in non-intuitive ways and it won't be easy to debug. Whereas in the Naïve Bayes model, you can look at the parameters and try to see what might be happening.

There, the problem to be solved was pretty clear, and main question facing me was "What is the minimum effort that I can put forth and get something into the hands of the users so that I can test whether the solution is useful?"

A big issue that impacts machine learning at startups is that manually labeled training data might be hard to come by. This is why at RelateIQ a lot of the models I had to build involved a lot of manual intervention. I had to be very careful about picking the right features that, based on my experience, would not cause me to overtrain; because I knew my training data was so biased and so limited that I could not rely on cross validation. I had to basically look at each feature and ask, "While using this feature gives me a reduction in test error, what is the chance that this is simply because of the way the data in the training set was selected?"

Another side effect of limited training data is that sometimes it is useful to closely examine and perhaps manually twiddle the model's parameters, setting them to values (or signs) that intuitively make sense. This intuition has to be balanced against the parameters coming from the learning algorithms. As in most cases, extensive experience with learning algorithms in limited data situations helps.

Yet another impact of limited training data is the high likelihood that the training data is from a different distribution as your deployment data. For example, at the beginning the startup might have 48 customers, and they are probably all friends of the CEO. The models trained on data obtained from these customers are likely to be biased towards them. However, one year down the line, the startup might have 4,800 customers. If one is not careful, the models created in those early days will fail miserably on the new customers a year later.

You said that you're building out your own tools that you took for granted at RelateIQ. That is really interesting because when you work in data science at a company, you have a lot of things that are taken care of for you. Deployment usually is handled by other people as a day job. So how are you approaching that? What are you rebuilding, and how do you know how to rebuild it?

I am playing around with some ideas on mobile, app re-engagement, and advertising. I am currently implementing the backend part. But since I learnt from RelateIQ that I need to invest in tools early, I am being careful in designing the backend "properly" from the very beginning. It plays nicely with my IntelliJ IDE; I am ensuring that I am able to run it entirely offline. When I push code to GitHub, the remote systems pick the code up, automatically compile/test it on the server, run the DB migration scripts, automatically deploy the APIs, etc. If we were to not invest in this now, then I would have to do this entire process manually, every time I deploy some small bug-fix. Moreover, I would probably make mistakes in the deployment steps (forgetting to migrate the DB) and find bugs that end up being simple deployment errors.

How do I learn all this? Some of this I worked on, but a lot of these technologies were just words that I heard while at RelateIQ. I was working with these extremely talented engineers, and I heard them talk about Docker or Maven or Guice all the time. So when I left and started working on my own company, I Googled all this stuff. Google, along with sites like Stack Overflow, is a great resource for these things.

I have been pretty shameless about asking people for help, because for any topic I am working on, except maybe one or two topics, there is someone out there who knows it better than me.

And if all else fails, there's GChat so I can ping my ex-colleagues who are friends of mine. These guys have so many years of experience that even without my completely describing the situation, they are able to point me in the right direction. I did the same thing when I moved to RelateIQ and I was the only data scientist. Since I couldn't talk to anyone at RelateIQ about some of the problems I was tackling, I would ping my friends from Yahoo! Research to ensure that I was thinking about the problem and solution the right way. In turn I would help them think through data mining problems they were thinking of. I have been pretty shameless about asking people for help, because for any topic I am working on, except maybe one or two topics, there is someone out there who knows it better than me.

If you want to start your own company, you have to build the first version of the product. Eventually, you'll get funding and be able to hire amazing engineers, who are going to scoff at my code and change everything, and I'm okay with that. But in the meantime, I need to build this. And there's never been a better time to build things yourself – there are a lot of good tools out there. You can use the Google App Engine for example, and many aspects of the backend are abstracted away. They have their own version of Task Queue, their own databases. You don't even know how your data is hosted. I didn't want to use Google App Engine because it abstracts way too much, and I felt like the lock-in

might be rather severe. That's why I am building the backend from scratch on Digital Ocean.

The other reason to build all this myself is that it helps me interview people. *[Laughs.]*

One day, I will have to interview someone who will help me with DevOps. It is significantly harder to interview someone if I don't know much about DevOps myself. My advice to anyone who's thinking of leaving a company and starting their own is that six months before you plan to leave, talk to your CTO or your manager. Tell them what your career goals are, and that you want to do this. Have them put you in situations where you can learn some of this stuff because there's nothing better than learning on the job. Someone's paying you, and you're learning while doing work for them. Like I said earlier, Silicon Valley is good at giving you opportunities to learn and extend yourself.

What are the opportunities you see in data science right now?

If you want to start your own company, you have to build the first version of the product.

There are many different aspects of data science. One is data analysis, to support business decisions. There is of course a huge need for data analysts in all sorts of businesses; at RelateIQ we need data scientists to analyze our product usage and SaaS business and suggest ways to improve the product or sales processes. But there is also a huge opportunity to actually build a layer between these data scientists and the data. These guys would prefer to use higher-level statistical languages such as R, but they want their analysis code to run on large-scale data on a distributed set of machines. There are a bunch of companies looking to provide this interface between the scientists and data, so that they write their analysis in R and don't have to worry about where it runs, and how it runs. The interface might even contain features that might help data scientists collaborate and make them much more productive. Mode Analytics and Sense are two of the newer companies I have seen in this market.

In terms of products enabled by data mining and machine learning, there are *huge* opportunities out there. Some are in the usual areas: Digital advertising, Search, and Recommendation systems. There are some mature players in these areas, providing both one-off services and platforms, but there is plenty of novel work coming out of startups as well.

One generic area that is seeing a lot of work is in trying to make sense of unstructured data. In the most straightforward cases, some startups are trying to automatically understand web pages and construct APIs over their data. However, at an abstract level this is what

RelateIQ is doing as well. RelateIQ is trying to mine people's communications data and give them insights about their own data. RelateIQ takes a mix of structured (phone call metadata) and unstructured information (email texts) and tries to extract structured objects that are of interest to the user (follow-up suggestions, new phone numbers for contacts, best connections to use to reach people etc).

While RelateIQ is mining relationship data sitting within enterprises, there is a huge opportunity to mine all sorts of unstructured data within enterprises and make it useful to them. For example, data within emails, calendars, etc. could be used to help large enterprises grow and maintain their talent; many startups are pursuing this.

Another area where data mining can help is by helping people deal with information overload. Right now there is all this news just flying by me. I always feel like I'm missing out on so much and so I need help consuming this. There are some companies trying to help with this, trying to use machine learning to do that. Have you seen Prismatic?

I've heard of them.

Given all this information, Google is in a position to completely personalize the web for me. Have you seen the movie "Her"? Other than the falling-in-love-with-a-robot part, why is the rest of that movie not a reality?

I downloaded the app a little while ago, and with some help from me it was able to deliver some relevant stories to me. However, it wasn't quite as relevant as I would have liked and it wasn't helping with the problem of me feeling that I'm missing out on a lot of good content, and, recently, I stopped using the app. Another potential example is Google that knows so much about me. If I was an Android user, they would know everything about my mobile use. I use Chrome, so they know about my browser use. I use Google Docs and Gmail so they have all my work data as well. I use a search tool heavily, so they have the set of things I am interested in as well. Given all this information, Google is in a position to completely personalize the web for me. Have you seen the movie "*Her*"? Other than the falling-in-love-with-a-robot part, why is the rest of that movie not a reality?

I think the technology to build much of that is already here. The data is siloed so maybe that's an issue. Maybe the user appetite to engage with the app in such a personal way is not there yet. I feel people may not be ready to give up that much control, but I think it is headed there. I think the next big thing is going to be in that vein. The way RelateIQ works for salespeople, there will be digital services that help regular people live. There's a soccer mom somewhere being driven insane by having to run her household, arrange for her kids to attend school and various activities, managing events for the entire family

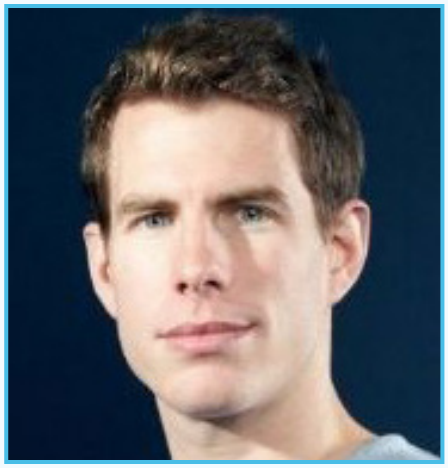
and interacting with her friends, all at the same time. Why is her phone not figuring all this out for her — making her life easier for her?

A dominant trend in the world is the move towards mobile. A lot of the mobile world right now is replicating what the desktop world did. On the desktop we have websites. And so we have a notion of apps on mobile. We used to navigate between websites, often through searches. So now we are devising a way to navigate across apps via deeplinks. However, it seems to me that my usage of my mobile device is very different from my usage of my laptop. I do most of my information access on my mobile device right now. I use my laptop for coding and for long-running information searches like buying something. It seems like technologies that make my life on the laptop easier may not necessarily work on mobile devices. I don't know the answers here yet, but I feel like data mining has a large role to play. This interests me a lot and I am likely going to select a problem in this space for my startup: a mobile frontend with an intelligent backend.

So there are plenty of huge opportunities; though, I think they are very difficult to predict and quantify.

SEAN GOURLEY Co-Founder and CTO at Quid

From Modeling War to Augmenting Human Intelligence



Sean is a physicist, decathlete, political advisor, and TED fellow. He is originally from New Zealand where he ran for national elected office and helped start New Zealand's first nanotech company. Sean studied at Oxford as a Rhodes Scholar, where he received a PhD for his research on the mathematical patterns that underlie modern war. This research has taken him all over the world, from the Pentagon, to the United Nations and Iraq. Previously, Sean worked at NASA on self-repairing nano-circuits and is a two-time New Zealand track and field champion. Sean is now based in San Francisco where he is the co-founder and CTO of Quid, an augmented intelligence company.

To start off, can you tell us a little bit more about your background from your early days to now?

For starters, I'm from New Zealand. Unlike many data scientists, I did not grow up doing huge amounts of mathematics, but I think this was probably a little bit of a conscious decision from my parents. When I was five or six years old, I used to stay awake all night under the bed covers with a flashlight, solving math problems . My parents thought, "He's probably doing more than enough math, so we don't need to push him on this skill." As a result, at school, I never really focused on math. Instead I spent a lot more time learning psychology, English, politics and philosophy.

I enrolled in university as a law student, which I really loved. After a few semesters, I also realized that law was a lot of hard work, and it was simply easier for me to get the best grades in math and physics. So after one year of university, I switched out of law and made the decision that I should focus on what I excelled at. I changed my major, but I made a deal with myself that if I didn't like it after a year of studying, I would go back and become a lawyer. As it turns out, I loved it so much that I went on to get a PhD in Physics.

So while I had a lot of mathematical abilities at a young age, I wasn't pushed in that direction. Instead, I spent a lot of my time learning about law, philosophy, politics and psychology. I truly believe that it gave me a better perspective on the world, than mathematics alone would have given. Although I didn't know it at the time, the combination of physics and politics would allow me to make breakthroughs in a field of mathematics that didn't yet exist.

I immersed myself in the world of physics and loved it. Mostly because it allowed me to ask questions about the world and come up with testable theories to explain it. With physics, you can explain why the sky is blue. It's fascinating to just be able to do something as simple as that. Once I got in further and started doing nanotechnology and quantum mechanics, I ended up pushing the boundaries of the world we experience. In nanotech, for example, you become obsessed with explaining the very small. The theories you develop start to speak to a world of atoms and electrons. The interactions don't make sense on a human scale; they're non-intuitive because you're not really modeling our human world anymore.

Although I didn't know it at the time, the combination of physics and politics would allow me to make breakthroughs in a field of mathematics that didn't yet exist.

Likewise, in cosmology, the equations you build represent systems on a galactic scale. Again, at the edges of modern physics, you're modeling worlds that are divorced from the everyday human experience. It's not really addressing the big questions that we face as humans. Questions like: "Why does the financial market move in a way that allows it to crash massively, while at other times remain stable?" "Why do wars seem to start?" "How do epidemics spread?" "Where do ideas come from and how to they evolve?"

These were the questions about our world that I wanted to answer — and I believed that the tools and techniques from physics and mathematics might have something to contribute.

It wasn't until I got to Oxford, being very lucky to go on a Rhodes scholarship, that I had the freedom to explore these ideas. I was originally taking my PhD in biomolecular motors, which, like most physics projects, involved a lot of time spent in the lab. After spending a couple of days in the lab, I thought, "I don't really want to spend the next five years in these rooms." I went looking around to see if there was a branch of physics that wouldn't involve time in a lab. As it happens, there was a really interesting professor there who was modeling the dynamics of human interactions, particularly financial markets. I asked him if he would take me on as one of his students, and after a bit of convincing, he said yes.

My supervisor's name was Neil Johnson. He was a relatively young physicist who was making a name for himself by publishing on a range of different topics, from quantum computers to statistical physics. I worked with him, getting my initial start in the field, by creating models for financial markets. For me, this felt like home.

The first thing I started researching was the dynamics of ensembles of agents. Or, simply put, what happens when a range of intelligent objects start to interact. We used this agent-based modeling approach to start to understand the dynamics of financial markets. Not necessarily to predict where the market was going, but rather, to understand the forces that are shaping and driving it. The work was pretty novel at the time, but it suffered from the limitation that the computer simulations being used to model human behavior could not capture all the intricacies of our human psychology.

This study raised some interesting thoughts. On one hand, there is the issue that we still don't fully understand the complexities of human decision-making. On the other hand, we're definitely more predictable than we think we are. What unfolded in the financial world was that humans got out of the market, and algorithms started trading. The algorithms looked exactly like our models. We had created pretty accurate simulations of a market of competing non-human algorithms, along with some warnings about volatility when algorithms dominate a market. This modeling of financial markets led me to my next line of research, and towards modeling the dynamics of insurgency.

War was topical in 2003, as the US had just sent a massive deployment of troops to both Iraq and Afghanistan. In 2003, we also saw the information landscape change as we started to get data sources, like blogs coming online, where reports of violence would be transmitted by many different sources, all of which could be read by machines. So not only could we create virtual models of insurgency, we could tune these models to precisely replicate the statistical signatures of the data that we were collecting in near real time. All of this required building machines that would read news and design algorithms that would extract events from these stories. This was a challenging proposition given the state of Natural Language Processing technology circa 2003. We used a lot of heuristic techniques combined with supervised machine learning models. They performed well enough that we were able to assemble a very complete data set of violent events. We analysed this data set for any statistical patterns and built the agent-based models to describe them.

You've got everything from data coming in, looking for signals within the noise, building models to replicate those dynamics, and being quite at the fringe of physics. In the end, my PhD was in physics, modeling the dynamics of insurgency mixed with some algorithms, natural language processing, and political dynamics.

That must have been a fascinating topic of study. In 2003, it sounds like it was quite hard to get the data to actually conduct analysis.

You could say that. The data that the U.S. military had was classified and as a foreigner, you weren't going to get it or even know if they had it to give. It was this restriction that actually drove us to use alternate data sources in the first place. We didn't think that our

open-source data was very good at the time. We thought that the classified data must be much better. It was funny, the first time at the Pentagon, I said, "Look, I've got this data, and this what we're seeing. If you've guys have better data, can I have it now?" After a few minutes of talking amongst themselves, they came back to me and simply said, "No. Your data is broadly in line with what we have. You don't need ours."

As it turns out, it wasn't just broadly in line — it was better! That was just crazy!

You see this transition from data being valuable to algorithms being valuable.

The idea that through open source intelligence, you can beat the entire US military's data collection about significant events in Iraq. When WikiLeaks released the Iraq significant events database, the information was of a lower resolution than the data we had. Data that was already out there and available to the public if you just had the right algorithms to make sense of it. This trend of open-source intelligence dominating closed data collection is one we are observing again and again. We saw it with the financial markets. It used to be, 'do you know the price and volume of stocks?' This was the valuable information. Then, price information becomes a commodity. So people switch to making sense of the data with advanced algorithms. You see this transition from data being valuable to algorithms being valuable.

I went through all that. It was a challenging few years. I spent time in Iraq, the Pentagon and the United Nations. I had enough war to last me a lifetime.

You mentioned spending time in Iraq. Was this during your PhD?

I wasn't in Iraq during my PhD. I went in 2008, after my PhD, but the events in the UN and the Pentagon were during my PhD. A lot of it was knocking on doors. I was literally in D.C. stating, "I've got this equation — all this useful data." I just showed it to a couple of people I knew, who helped get me in touch with some of their connections. Over the course of a week, I had impressed enough contacts that that I ended up in the Pentagon presenting to four-star generals, the intelligence team from US central command, and the Iraqi ambassador to the U.S.

What was the reaction like? How did the folks from the US government respond to your presentation of your research?

I expected two or three people to show up, but instead, I was surrounded by 40 people, circling one of those classic war room tables that they have in the Pentagon. Reflecting back, I would have prepared a more polished presentation if I had known exactly the

scale of my briefing. There were people who did not fully understand what the data was telling them, and were opposed to the findings. One of the main points of difference was that Pentagon analysts were insistent that there were six groups of insurgents in Iraq. Since I could not make any of the mathematical models work with six groups, they simply stated, "This is not how things are done. You don't know anything about this space." My rebuttal was that if they didn't like this theory, what was their theory to explain this data? They replied, "Our theory wasn't designed to explain any data," to which I replied, "How is that even a theory?"

It was as if I was talking to people who spent their whole lives avoiding numbers. They studied political science, and every decision they made along the way was so they didn't have to deal with mathematics or statistics. But there was the small percentage of those in the room who had spent time on the ground in Iraq, and seen the reality of the situation. When they saw the analysis, they understood what the numbers were saying and they agreed, "this explains a lot of what is going on in Iraq." The Iraqi ambassador to the US, who luckily had a degree in engineering, was able to see the power of the data driven approach. He said, "It's like the Wild West out there. There are hundreds of different groups fighting each other. We can't just sit down and negotiate a truce when the group might not even exist tomorrow. The models you have show this."

Did you find any allies amongst those you presented to?

Yes. People on the ground: the soldiers and the Iraqi government. Those were two big allies because they had the very real challenge of having to navigate this violence on a day-to-day basis. Many of the officers from West Point asked, "What do I do? What does this mean for me? How do I operationalise this? I've got guys on the ground and I don't want them harmed. What does this mean for getting them home safely?" You go through the dynamics with them; tell them what the basic statistical signatures are showing you, and what the models point to from a strategic perspective. Here is the probability of attack. Here's how the different insurgent groups coalesce. Here are the signals that suggest a group is starting to break apart, and operationally, they get it.

On the flip side however, many of the Pentagon analysts weren't moved. They would state, "We've got game theory." I replied, "What does game theory mean when you have hundreds of different groups constantly evolving and the estimated half-life of a group is under 6 months? What does that mean for your game theory?" The insurgents don't even know what's going on. How are they supposed to take the rational decisions needed for Game Theoretic models? Game theory is great if you're in the Cold War and have a good understanding of your one or two enemies, but these guys are 40 years behind. This is a different war.

Just as game theory emerged out of the research from the RAND corporation trying to model the standoff in the Cold War, are you saying that complexity is the new model for human actors and chaotic warfare?

That's exactly right. But they were limited by the restrictive nature of the classified worlds their analyst teams work in. If you were to ask them to look it up on Google, they would tell you, "I can't open Google. We've got an encrypted system. We've got to go into that room outside of the building to use Google!" They couldn't use any data that hadn't been approved. This limits their view of the world so much so that their analysis becomes dissociated from the activities on the ground.

That said, things are slowly changing. You see General Petraeus moving the thinking of the military towards a more data centric approach. David Kilcullen, an Australian ally, was chief adviser to Petraeus, and they started to become more data oriented. What they were doing with the data was still pretty naïve — but it was a start.

I think it's changed quite a bit over the last six or seven years, and it's more accepted that this is the way it's done. But the first time we submitted our research on the mathematical structure of insurgency to the top scientific journals, they said, "We don't do politics." To which we replied, "It's not politics, it's mathematics." Still, they were pretty closed to the idea of publishing this type of research. It took a long time. The academic establishment didn't want to know about it. The politicians didn't want to know about it. No one wanted to publish it. Not because it wasn't any good — but mostly because it didn't belong anywhere. So you had this new type of academic work that had no home.

In many ways, TED played a pretty instrumental role in getting this research into the public eye. They put me on stage in 2009. At the time, I was only 28. I was the youngest TED speaker from an academic background. Every other academic, who had been on the stage, had a very well established name in their field. Yet here I am presenting this work, I haven't published, and I've just finished my PhD, so everyone is like, "Who is this guy?" I was a nobody with some interesting ideas — nothing more. But that exposure (over 1 million downloads of the lecture) did force people to stand up and take notice. When we submitted the research the second time, the editors at *Nature* realized that they had to take a closer look at the work. The paper finally went out for peer review and was accepted. Later that year, the research was on the cover of *Nature*. It was a huge win for our work and for us. But it took three years to get to that place, after many of the world's journals said they wouldn't have anything to do with it.

Now, we've evolved the theory and so on, but the world also became ready to look at conflict in a way that was quantitative. The lesson is that you can have all the mathematics, you can have all the science, but you also need to bend the world. The world has to be ready,

but through telling great stories you can help it get there. The million plus viewers that downloaded the TED Talk, and the attention that it garnered, changed the conversation. It made people everywhere begin to think differently about war and mathematics.

The lesson is that you can have all the mathematics, you can have all the science, but you also need to bend the world.

For those six months from the TED talk until the *Nature* paper was published, I encountered a lot of criticism. *Wired* magazine put a out a very critical article about my work, saying, "This guy is naive. He's saying he's going to make war simple. But he doesn't know what he is doing," but they fundamentally didn't understand the research. At the time, I was surprised by the reaction to the research. I thought that the research would be relatively straightforward, that the reaction would be more positive and open, but there was so much politics surrounding this kind of research, it was always controversial. You can't expect to analyze an ongoing conflict and not deal with politics. At the time, I guess I saw science and politics as being two different things. Of course, fast forward a few years to today and the research is now widely accepted to the point of being "obvious." It's obvious and accepted that when people kill each other, it is done in a mathematically predictable way that doesn't seem to be dependent on politics or religion.

I still think that experience taught me a few things. One is, if you really want to change the way the world looks at things, you have to be ready to be the first one through the wall. You have to be ready to get the bloody nose that comes with breaking through entrenched ideas, and know that you're going to get beat up a little bit. The world won't accept a new idea without having you fight for it. The second is, that you get to write the story but you also have to be willing to tell the story. The third is, eventually, when the idea does come to be accepted, it will seem so obvious that everyone will forget there was any struggle to start with.

I remember coming out of that time and needing a break from academia. I started applying for jobs, not really knowing what I was looking to do, but wanting to explore some different options. I started looking for jobs in hedge funds, technology companies and the big strategy, consulting firms.

You did a stint at BCG — a consulting firm, right?

That's right. I spent a whole week there at the Chicago office. It was 2009 and during the recession. I felt like I needed a stable job, and I knew I wanted to get out of academia. But after a week I knew that it wasn't for me and I quit the job to move to San Francisco.

Before we move on to the story, I want to point out that you didn't have the usual graduate school experience, where you're tucked away in basement number 2, in building number 3, and you don't see anyone. You were out there trying to champion and defend this new idea.

Yes, and trying to get people to listen. That's exactly right.

Could you expand on the different ways in which your graduate experience was different? It's amazing that you had the audacity to go to D.C. and began knocking on doors, when most researchers would rather sit behind their papers.

In hindsight, I know that I was very lucky in that I managed to choose my supervisor very well. Neil Johnson gave me the freedom to succeed, and I think that was really important. I didn't have a predefined goal of what I was going to do when I started my PhD. I knew I was going to follow what was interesting to me, and I had the ability to do that. I think that's really important. You should pivot your research as you progress in your PhD, because in your third year you are simply going to know a whole lot more about what is interesting than you are in your first 6 months. Be open to finding that sidetrack that changes the direction of your path.

You're in a place, as a PhD student, to be able to think deeply about a problem and comment on things from a pretty unbiased angle. This is a very valuable thing, and something that should be encouraged more..

I think the other piece was the total time I spent in the physics lab department, which would probably turn out to be less than a few days. I didn't spend a lot of time in the physics department, but instead, I spent a lot of time talking to people in political science. I spent hours with the soldiers that were coming back from Iraq and starting their Master's in international relations. I spent a lot of time with people with a range of different ideas about how the world worked, and a lot of time reading interesting papers that were outside my discipline; collecting information and ideas from disparate places. I then assimilated this information together to create a new set of theories about war.

There are two very different strategies that I could have used to get my PhD. There was a strategy whereby I could work really hard in the physics lab, plugging away at a niche problem for 5 years and making an incremental gain. Or, I could expose myself to a range of different ideas that no one else from the world of physics was seeing. I could connect the dots better than anyone, and then put a structure on it to make it relevant to the world. That was very much my philosophy. I spent five years of my life asking questions and answering them, and if I'm going to spend that amount of time, the questions should be interesting to me.

Once you've taken care of your PhD requirements, you have a lot of freedom as a graduate student — probably more freedom than anywhere. Get out and enjoy that. Be a part of the conversation and seek to answer the things on your mind while always seeking out new things. Take risks!

Given everything you've just said, it actually seems like you had a great PhD experience, in contrast to other data scientists we've spoken with.

I loved it!

I ran track a lot; spending three hours a day training for a Decathlon, pole vaulting and hurdling. I think it was really necessary to do that physical exercise because it cleared my head every day. I never went to the track and came back with the work on my mind. If you clear your head every day, it allows you to have fresher thoughts and filters out ideas. Sleep and exercise are two things that we now know removes weaker synaptic connections. As you can imagine, I did a lot of sleeping and running and filtering out all the weak connections that I would make during the day.

Be open to finding that sidetrack that changes the direction of your path.

To be honest, I think I only did maybe 2 hours of work a day during my PhD. But it was five years of working for 2 hours a day. Everything else, like the conversations, the things outside of your field that you read, the random ideas you stumble across — these filled the rest of the day. You build a life around a space that will expose you to the maximum number of ideas, and you build a pruning system in your life that allows the concrete ones to stay. In that regard, you build a lifestyle, as a PhD, that is less traditional. The other life you can build is to show up at the lab, do your studies, and repeat. That's a lifestyle that's pretty well proven, but there's another, which is the one that's going to create the new connections between the dots.

I think that's what really excites us about putting this book together. We feel like this whole data science thing is comprised of people, like yourself, who are in the field defining and fleshing out what it means to bring analytics to industries that never had analytics before, and being willing to fight that uphill battle. Despite your unique experience, you decided you didn't want to stay in academia for the rest of your life. What made you decide to switch to this other world that you were creating?

I made that decision before the *Nature* paper came out. I was frustrated that you couldn't get the resources from outside to solve the kinds of problems I wanted to solve. I knew

we needed to combine teams together from different backgrounds. I knew I needed people with expertise in Natural Language Processing; infrastructure people to store and process large amounts of data. I would have loved to have had more marketing people to help spread the ideas.

Everyone asks, "Does your theory work?" and for me, I can tell you that it does, or I can show you it does. Showing is a much better way of doing it.

There were all these things I knew I needed, and in academia, all I had access to was grad students from the Physics Department. It was too limited. I couldn't get cross-departmental teams, and I couldn't run them on the scale I needed. Maybe after tenure I could have run a small team of 5-6 PhDs, but it felt too long to get there, and the team size seemed too small to solve the problems I wanted to solve. Physicists can do a lot of things, but they are not developers who are going to build a real time, self-updating database of Chinese television transcripts with high precision named-entity recognition engines. It was never going to happen. You need a database guru from EMC to build that type of infrastructure for you.

So academia was too constraining?

It was.

I remember a particular instance in one of my days at the Pentagon. There were a couple of guys from Lockheed Martin who were selling some sort of new radar tracking system. While we were both waiting for our respective meetings, I remember thinking, 'I'm on the wrong side of this equation. I'm here to give the Pentagon ideas. These guys are here to sell them product. They're going to get the money to build this — I'm going to have to hope that someone listens to me.' Somehow, the academics become the people that are the advisors, but the money is spent with the people who are the builders. Everyone asks, "Does your theory work?" and for me, I can tell you that it does, or I can show you it does. Showing is a much better way of doing it. For me, I have to make this thing real. I have to make this theory concrete. I had a real desire to build something that made use of this research.

The building side of it was key, and so I came out here to the Valley. That was a pretty big move. Honestly, I didn't know anything about business. I didn't even know what a Series A was. I didn't know anything about hiring, or legal, or product management and quality code. But despite all of that, I felt as though I was ready to start a tech company.

I think that's a good first step. You're starting at the bottom, but it's all uphill from there.

It was a good first step, but I still had to convince myself that I could start a company and take on this kind of risk. I remember coming out to San Francisco and sitting down with Max Levchin, talking about what it was like to start a company. He told me, "It's never going to be easier to start a company. You might think it's hard now and that's fine — but it will never be easier to start a company than when you've come out of graduate school, because you're living on no money. You've got no one to support. If it all goes south in a year, you're still incredibly employable and you can roll the dice." That stuck with me — "It's never going to be easier," because in my mind, I thought, "If I can go to McKinsey and learn all about business, when I go back, it's going to be easier." But that's not the case. It's never going to be easier to start a company than when you've just left grad school.

I think the second step was finding my co-founder, and without him I wouldn't have been able to make the jump into starting a company. He had been out in the Valley for about four years. He was the first employee of Yelp, and I thought he knew everything about startups and business. Of course, looking back, he didn't know everything, he just knew more than me, which wasn't hard, but the things he did understand were vital to our early progress and survival. He was instrumental in helping me build the infrastructure for the ideas and the product that I had. I could not have started the company by myself, nor would I have wanted to. Starting a company is difficult and trying to do that alone is too much responsibility, especially when it is the first company you have started. A co-founder makes it bearable.

What have been the biggest changes since grad school? I think you skipped the part where you tried to be employable, and you just went straight to employing people.

Well, I did try to be employable. I went around to the consultancies, and I remember they were quite interested in me. I ended up working at BCG, but I found there was no way for me to apply the kinds of research and theories I had developed to the problems that they were solving. Working at BCG, I felt as though I had lost the talents that made me unique.

It's like the Pentagon all over again, except you were going to be a part of the Pentagon!

That's right. I was looking at working there; trying to have it make sense for me, but it really didn't. The ironic thing is that now things have come full circle. We're selling software to all the major consulting firms like McKinsey, BCG and Bain. BCG and McKinsey now use our software, which is a great result, but at the time, I felt like I was a losing a part of myself by working there. I had worked on all these cutting edge ideas

about data, and I was asked to give them up so I could conform to the way things were being done at consulting firms. I did it for a week, and I finally said, "I can't do this." It just wasn't a fit for me.

I remember being at the BCG training program in little country resort just outside of Barcelona. It was a 3-week long immersion course for scientists and lawyers to give them business skills, or the "mini-MBA," as it was called. It was half way during the first week and I had had a call with one of the senior government officials in Iraq about the escalation of violence over there. The call ended at around 5am in the morning, as such I didn't get much sleep and I was showing up a half an hour late to the 8am training session in modern accounting practices for strategy, and the Partner running the training was obviously not happy with me. "This is very important that you show up here on time." Of course, they wanted (and were paying me) to come here and learn important skills for strategy consulting, but that just didn't seem to be so important as taking the late night phone call from the Iraqi government to discuss IED modelling techniques. At this point I was thinking that I might be in the wrong place. Right then I knew it wasn't for me and decided to get out at the first opportunity.

I was disappointed. I didn't want to be at BCG, but I didn't know where I did belong. I wanted to keep doing this research and push forward the data analysis techniques I had developed, but there wasn't a place to do it. There was nowhere that would employ me to do what I wanted to do. So, without any other options on the table, I decided that if I wanted to make these ideas real then I would have to create a company myself. I didn't know exactly how to do this, or what the company would look like, but ultimately I made the jump. I remember the day. It was a Sunday morning, and I was driving back from Los Angeles up the pacific coast highway. Somewhere right around Big Sur it became very clear to me , "I can't get on the plane tomorrow and fly back to BCG in Chicago."

I rang them up then and left a voice message, "I can't go back to work. I'm done." It was scary to do that, but it was also a rush to cut all the ties. I kept thinking, "I finally get to do this." That was the transition.

Wow. That's amazing. So after that point, you ended up creating Quid. Can you tell us about Quid?

Sure! I ended up out here in San Francisco in the middle of the recession without a job, without anywhere to stay, with only the last pay check from BCG in my bank account. That was when it all became real. I started to do some contracting work to pay the bills, just talking to companies that I thought had interesting data and they paid me to start playing with it to see if there was any value. At one of these companies, I met my co-founder Bob. He was the CEO of a company with some very interesting data, and I said,

"I can help you with that." We worked together for a few months, and it was a blast. So I told him, "You have to help me start Quid. We've got to start Quid. You have to get out of this company. Quit. You have to come to Quid!".

So we pitched the idea for Quid to Peter Thiel at a breakfast meeting at his house. He liked it and came in to lead the first round.

What was the pitch at the time? Were you pitching a commercial version of your research?

At the time, I didn't think my research was commercializable. I thought the military might buy it, but it wasn't clear that companies would buy the idea of an intelligence platform. No one was banging down our door asking for a platform to allow unstructured data, collected from outside sources, to make their biggest strategic decisions. It didn't seem like there was an obvious market. Customers wanted the machines to predict what to do next; to push a button and have the computer spit out an answer. But this wasn't what we were offering. We were saying that we could build an intelligence platform that would combine the best of the human brain with the best of the artificial, computer brain. We didn't have a name for it then, but today it would be known as augmented intelligence.

We needed a first group of customers who would adopt this new way of making decisions. We needed a group in which the decisions were too difficult for a computer to make by itself and a group in which the biological limitations of the human brain were running up against the increasing complexity of the world. Silicon Valley, in 2010, provided us with just this environment. There were groups of people, working to figure out the right M&A deals to make for big companies, like Microsoft and Google. Their task was almost impossible to do well, simply because of the time it takes to adequately understand an emerging technology space, before the space has changed to the point where your analysis is no longer accurate. This reminded us of the same challenges we had come across in Iraq. Trying to keep track of many small groups (startups), any of which could do damage to a larger dominant force (Google, Microsoft, etc.). It seemed like you could apply these same research techniques, developed to understand insurgents, to the global technology landscape. If you did that, you could make better bets than were currently being made. It could move the market cap of companies by billions and create new winners in the space.

That seemed like the right place to get started. In my mind, I thought, "I want to build this. I want to build a company that remotely monitors the globe and allows analysts to plug in and see structures, like I did in my PhD." By rolling out this technology to the corporate M&A market, it allowed us to begin the project with a clear business case on which to focus our development.

That's how we got our start, but in the back of my mind I was always thinking, "I have to hack this venture capital financing structure to do some really cool science fiction stuff. What I really want to do will take a long time, but as I go through this process, I'll be able to do it. First, I have to make money. Once you make money, you can do anything you want." This was the hack.

That's how we got our start, but in the back of my mind I was always thinking, "I have to hack this venture capital financing structure to do some really cool science fiction stuff. What I really want to do will take a long time, but as I go through this process, I'll be able to do it. First, I have to make money. Once you make money, you can do anything you want."

That being said, venture capital is not really set up to do this. For the most part it is easier to arbitrage market opportunities than it is to create science fiction type products. Even if you do get money for this, there's no guarantee that it is even possible to build the things you are imagining. Looking back now, 2009 was too early for this intelligence platform to exist. There were just too many technology solutions missing from the equation; too many things that we would have to build ourselves. The right time to make this for venture capital would have been at the start of 2012. I think if I wanted to start the company, I should have started in 2012, but the problem would not have been nearly as interesting to work on. This is the issue with venture capital — if you want to really push the limits of what science can do, there just might not be any business applications ready and waiting for you, once you've done it.

When things come to market, they are not as interesting as they were five years earlier, and when you live in academia, you're 10 years ahead of the market. You think a certain thing is possible, and of course, it's not possible for 10 years. One of the heuristics is that whenever a group of papers are published, it will take 10 years for any academic breakthroughs to become commercial realities.

You learn to appreciate that timeframe. You can't port your research straight into venture- at least not with the current financial tools that we have. If you've finished your PhD, and you can immediately start a company, then your PhD wasn't any good, because you should be far enough ahead of the world that there isn't a market for what you're building. On the flip side, you're going to be at a place where you're the first to market, and you've already made all the mistakes, and now others can copy your breakthroughs. That's a difficult place to be. It's an exciting place, but a difficult place from a business perspective.

In many ways you have to constrain your science aspirations by business realities. We had done this war stuff, and we were going to create a platform to improve private equity

investment decisions. It's not quite revolutionary, but you also have to keep in mind that it is the first step, and venture capital is all about levelling up. If you make the first step, then you move from $2.5M in funding to $10M. Make the next step and you get $50M to play with. You have to keep the grand vision, and at the same time execute on the day-to-day elements of creating product market fit.

My vision with Quid is to have an intelligence platform that monitors the world through open data sources, so that anyone can plug into the platform and see the patterns that shape the world. From that vantage point, everyone can make better decisions in their world that will ultimately impact our world. Users can see further, see deeper, leveraging deep intuitive AI combined with immersive visualizations. To ultimately amplify intelligence through software — that's where I want to go with this. I want to have a system that makes us smarter, that is distributed as widely as possible, and to as many people as possible.

We're talking about augmenting human cognition across the planet. I think we're maybe 10 years away from having a complete working version of this. There's certainly another billion dollars of investment here. Whether we get there or not, no one knows. But I think something to take away is that with science, you hold a piece of the puzzle, you direct the field, and you get to shape it a little bit.

Here, the traditional thinking is that if you don't win financially, you haven't won. But I know that by even being here, I've shaped the direction of the technological vector. I will continue to shape it as long as I keep playing. Even if I don't win the money at end of the game, I'll do very well, but the direction is going to be shaped because of me. In science, we know that we play a part in a bigger human endeavor and in the Valley its all about me — and did I win.

This is why we need to be in this game, because these things that are happening are shaping all of us, and if it's just a monetary grab, we're a little off. Science brings in the aspect of having a sense to do something worthwhile and good.

In 2006, you were working on this paper and *Nature* wasn't ready, the Pentagon definitely wasn't ready. Now, it's almost 10 years later, and now we see the movement in industry towards making strategy decisions. Relative to your paper in 2006, do you think that within that 10-year deadline, we're going to get to the place where your vision is?

The paper came out in 2009, so there's probably still a few years left before we see commercializations. Based on the work from 2006, we are definitely going to see this in the commercial world. In the last 6 months, since Quid released the second version

of our Intelligence Platform, we have seen a great uptake in all types of people using it; from hedge funds, to strategy consultants and advertising creatives. Fast forward another 18 months, I think that we will have thousands of strategy consultants running Quid software. I'm very confident that we're putting the elements in place to make this happen: the relationships, the technology, and the algorithms. We're getting our first course at the University College of London's management science class and the entire class is getting Quid accounts. They're plugging in. They're training on it. The first McKinsey teams and the first BCG teams are spinning up. We just signed a major deal with a group of publicists to roll this out to their creative strategists across the entire firm.

We've got small, boutique consultancies that are winning massive contracts because they're running Quid software. They're getting great results and even moving into new offices based on all the new business they're winning. Consulting used to be a case where you arbitrage smart PhD students and then you plug them in to BCG. You throw people at the problem. Now, these same people can drive the kinds of software Quid is pioneering. The AI behind the Quid engine can do a lot of the heavy lifting, saving literally, weeks of work. The analysts can spend time doing the things that matter — which is figuring out the implications of the strategy in front of them. Hopefully, if nothing else, I've given some of our users a bit of their life back instead of spending it doing those tedious things.

A lot of the process, for me, is continuously learning to let go of the things that I have built and let other people in the company take ownership of them.

The strategy part of the problem is well on the way towards being solved, and I think the next step is to start integrating more of our Artificial Intelligence capabilities into the system. Scientists talk about the brain having a neural circuitry for intuition. Expert intuition comes from two parts of the brain: the precuneus and the caudate nucleus. The precuneus is the pattern recognition engine that identifies patterns to extract signals, where most of us non-experts would only see noise. I think we've built a lot of that in the current release of the Quid platform. We've built a lot of pretty good pattern recognition engines, to allow users to see structure in news, science, Twitter etc.

The next phase of the Quid project is to build the AI version of the caudate nucleus. This is the part of the brain associated with the learned response function. I think in the next couple of years, we will have built a good, learned response function into the Quid platform. At the moment, humans are learning the patterns and figuring out what they should do next based on their experience. We can start hinting at things that have happened in the past. That's not to say that we should replicate them, but we can build

on that functionality so they can see what an artificial caudate nucleus would look like on top of these data streams. It's a kind of simulation that would project the current world forward in time to explore different scenarios. I think that's going to be really powerful.

I think what the Quid platform ultimately does is allow people that are good at recognizing patterns, and knowing what to do about them, to jump between different areas of expertise. You should be able to jump into the world of material science and see all the papers written in that field, and then jump into the world of swarm robotics for UAVs, and to then into the world of Ukrainian politics. You should be able to leverage the intelligence engines of Quid to give you a deep understanding, across all these fields, in just a few hours.

I think when we put that tool in front of people, they will begin to broaden their knowledge. They will see further and they will see new things. I want to try and replicate the experience I had in grad school, where I was able to jump between things but to give that in a tool to other people, and I think that will change the way people will see the world. I think it will also allow things to be thought and to be built in ways that otherwise wouldn't have happened. I'm more confident now than ever of getting that right.

A lot of the process, for me, is continuously learning to let go of the things that I have built and let other people in the company take ownership of them. It seems obvious, but it's a continual process of letting go. You build the first versions and then you have to step back and give it to the other team members. You might not like how they choose to update your solution, but the point is that it's no longer just yours anymore. The challenge then becomes knowing when to let go and when to step back in to make sure the essential parts of the project are still being maintained.

For example, one of the key features of the product is the animated transition between data spaces. When building this, I was adamant that we had to have transitions. Some other members of the team rightfully challenged this because it's an expensive part of the product to build. They would ask, "Why do you need transitions?" There's no real empirical evidence as to why we need this feature, but it somehow just feels right. So I replied, "We've got to have transitions. The data has to move between a timeline and a network. People need to orientate themselves". Other members of the team again objected stating, "No one wants to have that." Until I finally stated, "I hear you, but we're doing it."

Fast forward a few weeks, and one of the engineers says that a cylindrical coordinate transformation would be better. I said, "You know what? I never thought of that. I'm not sure it does." Then I looked at it, and they were right; it does. A cylindrical transformation

— translating between two points using cylindrical coordinates; I never would have thought of that. I had nothing more to add. Their understanding of this problem was beyond me. That was the right time to let go.

You have things that are important and then at some point, you realize, "I have nothing to add to this." That's the continual balance when you create things that are pushing the boundaries of what is possible. It's important to know what matters and what doesn't.

That's the beautiful lesson in leadership. It must have been a hard lesson to learn. I feel like there's so much investment involved. Not only have you carried this, you also fought for it for so long, and then, at the very end, you give it away to others torchbearers.

That's right, and it can only be done with thousands of people building this. It's fighting the battles that need to be fought and letting go of the ones that don't. Now, for me, it's trusting in the machinery that I've set up, and it's also going out and helping the world to learn to use this. With a team, you can keep making the products better.

In life, it's always 0 to 1, and you're either a one person or a 1 to 1.1 person. I think a lot of things we do in life are 0 to 1, and that's also important to recognize. That means that when you're the first one through the wall, you have to fight for it. No one wants to listen to you, and by the time the idea gets accepted, you're bored with it. You move on to the next thing.

The nice thing about startups is there's always a next thing. Everything you do is something you're doing for the first time and not something you're very good at. It's something you're unqualified to be doing.

Now at this point in your life, I have to say that you have a lot of credibility in doing things you're unqualified for.

That's how you learn about your strength. My strength is consistently doing things I'm not very good at and quickly becoming reasonably competent. My hope for grad students coming out of their PhD is to pick whatever job you want in a way that takes advantage of the skills you have as a graduate student. In data science, if you feel you have to conform too much in a box and don't get the freedom, that job is not for you. Find a place where you can shape a little bit of the world and keep doing that, which will probably involve doing a lot of things that you're unqualified for and not very good at.

I think we've got a very narrow view of what data science is, which has largely been shaped by data analysts working in the big social networks, like Facebook and Linkedin.

So for many of us, data science seems concerned with things like A/B testing to optimize personalized ad recommendations. But data science can be so much more than this. We must recognizing what data can do, what data can't do, recognizing that it's messy, that it's biased, and understanding that it needs a human layer — that it needs stories. Recognizing that it can solve a certain degree of complexity, but it can't solve any further. Remember that humans have biases and they absolutely need data. We don't want to move towards naive empiricism — that's not what data science should be. It's not what science teaches us, but, at the same time, we don't need to throw data out the window just because it can't push a button to solve an equation.

> *In life, it's always 0 to 1, and you're either a one person or a 1 to 1.1 person. I think a lot of things we do in life are 0 to 1, and that's also important to recognize. That means that when you're the first one through the wall, you have to fight for it.*

I think a lot of that will come together. Data science will evolve. I think the second piece is data scientists have an obligation to do good things in this world with that data. It's not enough to just not be evil; it can fundamentally be good. If you come out of science, you are contributing to the world's knowledge. When you come to the business world, you should also be contributing towards building the tools that help us to live and function in society.

This is an imperative that should be fought for very hard. This should be one of the decisions you make in the jobs that you do. We have a set of technologies that can and will shape our world in ways that are positive and potentially very negative. It ultimately comes down to the mentality of the people that are building the technologies. We can wash our hands and say, "I can't do anything about it. It's not in my control." But we really need to challenge this assumption. You can do something about it! You're making this or your company is making this stuff. You're the data scientists that are building this stuff. Of course you can do something about it, and of course you have that responsibility.

If you choose to do it in a different way, you are the one shaping the world. We are the people who are creating this technology, so you can't just wash your hands and say you're not part of it. We saw what happened when a bunch of quants on Wall Street, with little regard for the consequences said, "I'm just going to use these algorithms to make money." This is not good enough. You can't arbitrage a system and make money for yourself without also having the responsibility to make it better.

So much of data science has been concerned with equations for the optimization of an existing world. But we need to use data science to build and engineer a better world, and that's where it starts to move beyond black box predictions and basic statistical

tools and moves into design. One of the big things for data scientists is to understand that their role is also one of design. If you create algorithms, you shape the behavior of people who interact with these algorithms. So what kind of behavior are you designing?

I think data science is really going to become more of a product design process; actually an algorithm design process. Algorithms take information and direct us; whether it's the information we read, the music we listen to, the places we drink coffee, the friends we meet, or the updates in our lives.

Data scientists have an obligation to do good things in this world with that data. It's not enough to just not be evil; it can fundamentally be good.

You are designing algorithms that fundamentally shape humanity, and we do it in on a population scale in the billions. So how we choose to shape this world certainly has a lot of challenges. We can't just hide behind the imperative to optimize an algorithm for maximum revenue. You designed an algorithm that created a certain kind of behavior — for better or worse — and now this algorithm is potentially impacting the lives of billions of people you have never met. What kind of behavior do we want? I think you need to fall on the line of making humans more human, making them see further, making them see deeper, making them understand and appreciate the nuance. Don't try to hide the complexity from them, but instead, make them more conscious. Make them smarter. Help make them smarter. I think that's what you design for. That's what you use data science for.

That is amazing. I love that vision, the imperative you stated for people that want to work in data science.

I think most of the data scientists I meet are pretty good people. It's been refreshing to see this culture emerge from the community that we built up over the years. It's one of the things we started in 2009, DDG (Data Drinking Group), which was Pete Skomoroch, Mike Driscoll, DJ Patil, Bradford Cross, and me. There were five or six of us that would just get together and drink and talk about this stuff. These were some of the most creative times. The ideas we discussed, the questions we would ask, and the technology we would share. I think this really influenced how the discipline is unfolding today.

Data science is its own philosophy. We're not the same as the production engineers of the world. We're not product people. We're our own kind of group with our own set of values. Data scientists have a distinct kind of DNA that is measurably different from a lot of the other groups. As this culture emerges and develops, I think good things will come from it. I've always been very impressed by the data science people I've met. They have this nice coupling between the real word and the computer world. They know data and

they know engineering. They value the beauty in their algorithms and the beauty in their design. They span a lot of traditional disciplines and can combine their experiences to create new things. I think we're going to see this group of data scientists solve many of the bigger problems we are facing in the world, and that's pretty exciting.

JONATHAN GOLDMAN

Director of Data Science and Analytics at Intuit

How to Build Novel Data Products and Companies

Jonathan is currently Director of Data Science and Analytics at Intuit. He co-founded Level Up Analytics, a premier data science consulting company focused on data science, big data and analytics which Intuit acquired in 2013. From 2006–09 he led the product analytics team at LinkedIn which was responsible for creating new data-driven products. While at LinkedIn he invented the "People You May Know" product and algorithm which was directly responsible for getting millions of users connected and more engaged with LinkedIn.

He received a Ph.D. in physics in 2005 from Stanford where he worked on quantum computing and a B.S. in physics from MIT.

Can you give us a sense of the background and the path that you've taken to get where you are today?

I completed my Bachelors in Physics at MIT. I just absolutely love math and physics. I actually loved a lot of other fields as well, but knew I wanted to stay with math and physics in particular. I also absolutely loved MIT — it was the perfect place for me. When it came to graduation, however, I still didn't know what I wanted to do with my future. I knew I wanted to do something more in science, but I didn't know if I definitely wanted to be a professor. I ended up applying to Ph.D. programs but still wasn't certain if that was what I wanted to do.

I also applied for a few jobs, but was just not excited about any of the jobs I saw, and how they would leverage my skills. In comparison, grad school was exciting since I would get to work on fundamental research there. At the time, I was really excited about what was happening in the world of quantum computing.

I got into Stanford, and I found an advisor who was specifically working on quantum computing. So I came out to Stanford and liked it for a while, but towards the later part of my Ph.D. recognized I wanted to something else. Research was hard and not as rewarding in the short-term — it took me seven years to get the results that I needed to graduate. It was in my fifth or sixth year that I thought, "I want to do something that has a little bit more immediate impact."

The parts of the Ph.D. program I loved most were when I was actually getting the data, analyzing it, and iterating very fast. I had these experiments I'd have to run for 30 hours, and basically after that, the system would shut down, restart my experiment, and it would take a day or two to get the system to reset. It was during this period that I was getting this amazing data, make a hypothesis then and test it. I loved the actual thinking, the theoretical aspects of it, what that told me to do with the experiment, and what parameters to explore.

Towards the end of my program, I got involved in some entrepreneurship activities at Stanford. I got involved in this organization called a nanotechnology forum, where Steve Chu, Stanford physics professor and later the Secretary of Energy came to speak. A lot was happening back in the early 2000s in that area. I was trying to go into that area, looking at solar energy technologies — I was very excited about that. But then I looked at a few of the solar technology companies, and the basic approach that they had was, "Hey, you get to work on this technology as a postdoc, and if it works, you'll get a full-time job. If not, that's a nice postdoc for a year or two." That just didn't seem appealing to me.

At the end of graduate school, I was looking for a job, and I knew at that point I just did not want to stay and do a postdoc. I ended up going to the consulting firm Accenture, and I was excited about going to work in energy. I had been working on energy-related stuff, and I was getting more excited and interested in that. I wanted to work in strategy for Accenture — the focus was in the utility/energy sector, especially in the natural gas market.

Academia becomes a very competitive world since you have to make a name for yourself to succeed. The business world is also competitive, but in my experience, teamwork is more highly valued there because it really does take a significant effort from many people to make something interesting happen.

So, I was working for a little while on natural gas strategy for one of the partners, and that was fun. I got put on a project to work at a utility company, and it was good to get that exposure — to find out what the corporate world is like. What is it like to do consulting? What's it like to work in this company? How do they operate? I actually learned a lot about how to communicate and how they work; it's such a different world from academia.

Can you tell us a bit more about what you did at Accenture? What were you involved in there?

I was in the supply chain project for a utility company, and we did a lot of work on supply and demand, and other sorts of optimizations. When should the utility company

buy? How much inventory should they have and what do they have to plan for? They're interesting problems because you need math and analytics to figure out the optimizations. In the case of a utility company, it's different because you have to plan for worst-case scenarios. If there's a storm, I need to be able to repair everything quickly enough so people have enough power. There's demand and supply planning, as well as strategic sourcing — there's a whole bunch of interesting problems.

Can you tell us more about your transition from Accenture to LinkedIn?

At that point where I was thinking, "Let me see if I can do something a little more technical." I felt like I learned what I needed to learn so I was trying to find new projects. I started looking around to new places, including LinkedIn. Initially it seemed like it was a recruiting platform and I wasn't that excited about it, but after I went and met with various people there, learned about their data, and learned about what they were thinking about, I thought "Wow, this is awesome."

What was it about LinkedIn that hooked you?

Well, what really excited me was thinking, "Well, look, you have this data about people's careers, where they went to school, where they are now working, what they have done in their careers, and descriptions of their past jobs. So how do I help people get the right job?" It's a problem that actually felt very personal. While I'm trying to find the right career for me, I could help work on solving that problem for others at scale.

The data was all there, and I could ask questions about the data very quickly. It was exactly the part of the Ph.D. program that I liked. Suddenly I didn't have to deal with the experimental apparatus which took me two years to build. It was like, boom, I have the data, and it's actually very interesting. I was learning all these new techniques and it was great.

Within two weeks of starting, I had already felt that this was my dream job. It was awesome, and I totally loved it. I found people even more collaborative in companies than they were in university research — we were all working to help the company do well and make a dent in the universe. In academia you also try to make a dent, but it was very often your own dent. Academia becomes a very competitive world since you have to make a name for yourself to succeed. The business world is also competitive, but in my experience, teamwork is more highly valued there because it really does take a significant effort from many people to make something interesting happen.

It sounds like you really enjoyed your time at LinkedIn. What did you do there?

I was trying to figure out what I could do with the data to improve things. One project I

worked on involved sending invites on LinkedIn. I looked at questions such as whether or not the click-through-rates changed depending on the level of the person who sent it to you (more senior than you, more junior to you, a peer). Something else I thought of and looked at was the reminder emails that we send a week or two later after you haven't accepted an invite. I looked into the best time to send such a reminder, and discovered neat facts such as 80% of invites go to people in the same time zone. This means that even though I don't know what time zone you're in, I can guess pretty well from the time-zone of the person who sent you the invite. We optimized the time of day that the email went out and saw boosts of 2-3% in click-through rate. This improvement compounds and the result can be massive.

Basically, we were trying to look for all these little knobs to turn to understand the LinkedIn dynamic and to understand LinkedIn at a fundamental, physical level. I thought of it as a physics problem involving people and invites. I was asking myself — who's connected to whom, and how can I get more people to join? How can I get more people connected? When you understand the system, you try to think of it not just as these disparate things but more as an overall global pattern that you want to understand — an engine that you want to get to move faster.

We optimized the time of day that the email went out and saw boosts of 2-3% in click-through rate. This improvement compounds and the result can be massive.

I started thinking about some of the dynamics involved in what gets people to sign up, and then I also started looking at the data. I found that a lot of people didn't even have that many connections. And people were not going to really get the value of LinkedIn until they had a good network — until they had ten, twenty or thirty connections. Most people only had one, two, even zero. I observed these things in the data, and realized that we really need to just work on getting people connected. I asked myself — how can we get more people connected? Well, we can make it easier for you to find people to connect with. Back then, there was Friendster, MySpace, and the beginnings of Facebook, and no one had been working on recommending people you might know.

Steve Stegman (Steve was what we would call a data scientist today) and I, within the span of one day, conceived the beginnings of the "*Viewers of this profile also viewed…*" feature. We could quickly get stuff out onto the site, test it, and see the click-through rates, so that was awesome. I had this idea of trying to recommend people you know and we ultimately called it "People You May Know". I was working on the heuristics, mostly at night, just iterating and iterating, and asking, "*What are the things that can work?*" And we ended up using a lot of stuff like company and school, and also the graph structure of how connected they are. The initial click-through rates were amazing — and then

machine learning helped increase click-through rates another 2- to 3-fold. This work was spearheaded by Monica Rogati who I hired onto my team.

Basically, we were trying to look for all these little knobs to turn to understand the LinkedIn dynamic and to understand LinkedIn at a fundamental, physical level.

This was not a product that was on any roadmap — I think that's an important thing to point out. I pitched "People You May Know" to a few product managers and they were all lukewarm about the idea. It was hard finding people who really bought into the idea at the beginning, but we ran tests and had data we could go back to and show people. Once we had data, no-one stopped us from expanding and doing more but it still took some time to get the proper engineering investment we needed. Because of the viral nature of "People You May Know", we demonstrated with data that this feature got millions of users back to the site who otherwise would not have visited the site. We showed this to Jeff Weiner in 2009, and he was like, "Yes, we've got to go on and do more." At that point there was lots more engineering investment put in place across LinkedIn and fortunately PYMK got significant additional investment.

This was a great example of a data product that was never actually on the product roadmap. It's the impact that a data scientist can have on a business, because you can observe some pattern in the data, build something, and start doing some pretty sophisticated stuff with all these different signals. You end up transforming the trajectory of growth.

"People You May Know" started as my original work. I did basically all of it initially, including the algorithm and the product, but ultimately, as it grew and grew, many more people became involved. Monica and Steve Stegman made contributions to some of the algorithm, and DJ helped with getting it onto mobile and getting it faster. Other product managers, like Janet, were also involved.

Later on in your career, you started your own company with your wife and a third co-founder, Lucian Lita — can you tell us more about this? What was it like transitioning from a role as a data scientist at a large company to running your own?

The three of us saw this opportunity — the demand for data science and building technology that would help solve data science problems. We saw a huge need that was just constant, and thought we could build a premier consulting firm and we would go to these companies and help them transform their businesses, while hiring people that we really liked working with.

The amazing thing was that we were able to get really good talent, get really good clients, and work on really challenging problems. There weren't that many people doing exactly what we were doing — no-one else did the full end-to-end, including "What's the business problem you're facing? Where's the place we can have the most impact? What technology might need to be built or deployed? What algorithms and analysis need to be done? We could do the full stack — I think a lot of companies really liked that approach.

One of our clients, Intuit, after we got to know them and they got to know us, approached us about getting our entire company focused on Intuit — namely they wanted to acquire us. We really liked the problem they were working on. They were fundamentally changing people's lives by making it easier to manage their finances, do their taxes and run a small business. It's actually quite an interesting problem because they see so much of the economy. They are really truly one of the few companies that I think is mapping the world's economy. You could say that LinkedIn is mapping the talent economy, but Intuit is actually mapping the real transactions that are happening. I don't know any other company that has such interesting data. The impact on the economy and economic wealth is profound. To me, it was a good mission to be a part of, and I really liked the culture and the people.

Given your own experiences in a PhD program, what advice do you have for our readers who are in a PhD, or just recently finished one, and are looking to start their career in data science?

I think one of the most important things is to learn to be curious.

Find the companies that are aligned with your values, where you get to work on things that are impactful and making a dent in the universe. There's never going to be a shortage of interesting problems to work on that are massive and impactful. When you're at that kind of company, it's easier to take that data and turn the data into transformational business impact.

I think one of the most important things is to learn to be curious. You see something that might spark new questions for future projects. Once you're curious about something with the data, you'll figure out how to go solve and answer those questions, regardless of the technique. You need to be able to go back and forth in an iterative manner as businesses don't always have well-defined problems.

WILLIAM CHEN Data Scientist at Quora

From Undergraduate to Data Scientist

William Chen is a data scientist at Quora, where he helps grow and share the world's knowledge. He became a data scientist after finishing his joint degree in Statistics and Applied Math at Harvard, and is part of the first wave of college undergraduates who took data science courses and sought data science jobs straight after graduation. Prior to joining Quora full time, he interned at Quora and Etsy as data interns. He has a passion for telling stories with data, and shares his knowledge extensively on Quora.

William is one of the co-authors of this book.

Can you tell us about your journey transitioning into data science?

I started my freshman year at Harvard wanting to study math, but then took Stat 110 with Joe Blitzstein. The class changed the way I thought about uncertainty and everyday events, while teaching me how to value intuition and communication. The class influenced me to declare statistics as my major in my sophomore year.

In my sophomore year, I started looking around for internships that would use some of my probability and statistics background. My knowledge was mostly theoretical then with little experience in application, so I was pleasantly surprised when Etsy invited me to intern with them as a data analyst. This was my introduction to using data to improve business — every facet of my internship helped me grow and develop my skills as a budding data scientist.

Etsy is a very metrics-driven company and I was able to see and understand the heart of how Etsy makes decisions with A/B testing. The frequent statistics discussions on the mailing list were engaging and I was able to learn about common techniques and potential pitfalls in metrics-driven tech companies.

The presentation of data at Etsy was beautiful (with d3 dashboards and highly polished slide decks). In that kind of environment and attention to visuals, I taught myself ggplot2 and started making my own plots and graphics. I was able to learn a lot during that internship — it was the start of my career in data science.

After my internship at Etsy ended, I started my junior year. That year, I returned to being a teaching fellow (the equivalent of an undergraduate teaching assistant) for Stat 110.

In helping people with their probability problems, I realized that teaching probability helped me improve both my communication and storytelling capabilities. It was also very enjoyable and I got more in the habit of trying to share and teach whatever I could.

Not having enough programming background to implement your statistics knowledge can severely limit the number of things you can do. I took computer science courses so I could more effectively do statistics.

My junior year, I also started to take a lot more CS classes as I realized their importance in a data science role. Not having enough programming background to implement your statistics knowledge can severely limit the number of things you can do. I realized that having both was imperative to succeeding in a data science career, so I worked to excel at their intersection by taking classes that I felt would augment my skills.

I was also applying for internships my junior year, with the mindset that I wanted to use my statistical and programming skills to help companies make better decisions. I received an internship offer from Quora and decided to take it, even though I was still fairly new to the product at the time.

At Quora, I touched a lot more of the codebase and learned much more about software engineering. There was a sense of dynamism and importance to my project. It involved new growth initiatives, and I appreciated the level of freedom and trust that Quora gave me. I enjoyed my time there working with both the people and the product a lot, so I decided to go back full-time.

In my senior year, I continued developing my statistical and programming toolkit while working on my thesis.

Why did you choose to major in statistics instead of computer science?

I put a lot of time into Stat 110 and a whole bunch of other statistics classes — I enjoyed those classes so much that it would have been unreasonable for me to major in anything else!

During my internship at Etsy, I saw first-hand how limited my abilities would be if I could only do statistics and not code. I put a lot of effort that summer into developing my abilities to analyze data in R.

My junior and senior years I took an equal load of statistics and computer science courses.

I took computer science courses so I could more effectively do statistics. I took classes to make me better at applying statistics (Machine Learning, Parallel Programming, Web Development, Data Science) or just because they were fun mathematical topics (Data Structures and Algorithms, Economics and Computer Science).

My primary interest is still statistics, but I heavily value computer science since it empowers me to do more complicated analyses, generate visualizations, deal with massive amounts of data, and automate away a lot of my work so I can focus on what's really interesting.

That being said, I actually declared a secondary (aka minor) in Computer Science my Senior Spring. I fulfilled its requirements (accidentally) and pursued the secondary because it would require no extra effort on my part, just some paperwork.

Can you tell us more about what you felt that your main challenge was during your data internships?

I realized that I needed to prioritize things that would have the most impact for the company.

One exciting thing about working for a tech company where data is central is that there's so many potential projects to tackle. There's so much data that can be analyzed and never enough data scientists to really look deeply into every single thing. My main challenge during my internships, especially at Quora, was figuring out how to prioritize all the possible things I could be doing, especially since I took on many projects in parallel.

At Quora, I realized I couldn't replicate what I did at school by working on everything at the same time. I realized that I needed to prioritize things that would have the most impact for the company. I spent a bit too much time working on certain tools and not enough time focusing on researching growth initiatives that would have potentially higher impact.

How do you see data science in terms of it being the intersection of math, statistics and computer science? What weight would you give each in terms of importance?

I would say that the programming and software engineering part is very important because you may be expected to implement models, write dashboards, and pull out data in creative ways. You'll be the one in charge of hauling your own data. You'll be the one who owns the end-to-end and the full execution, from pulling out the data to presenting it to the company.

The Pareto principle is in full effect here. Eighty percent of the time is spent pulling the data, cleaning the data, and writing the code for your analysis. I found this true during my internships (especially because I was new to everything). A good coding background is particularly important here, and can save you a lot of time and frustration.

To emphasize: pulling the data and figuring out what to do with it takes an enormous amount of time, and often doesn't require any statistics knowledge. A lot of this is software engineering and writing efficient queries or efficient ways to move around and analyze your data. Programming is important here.

One interesting thing to note is that the statistics used day-to-day in data science is really different than the kind of statistics you'd read about in a recent research paper. There's a bias towards methods that are fast, interpretable, and reliable instead of theoretically perfect.

While the statistics and math may not be that complicated, a strong background in math and statistics is still important to gather the intuition you need to distinguish real insights from fake insights. Also, a strong background and experience will give you better intuition on how to solve some of your company's harder problems. You may have a better intuition on why a certain metric might be falling or why people are suddenly more engaged in your product.

The better you understand the theoretical bases around a certain idea or concept, the better you can articulate what you're doing and communicate it with the rest of your team.

Another benefit of a strong statistics and math background is the contribution to communication. The better you understand the theoretical bases around a certain idea or concept, the better you can articulate what you're doing and communicate it with the rest of your team. As a data scientist, a large portion of your work is presenting an action that you feel would have an impact. Communication is very important to make that happen.

Some data science roles require a very strong statistical or machine learning background. You might be working on a feed or recommendation engine. Or dealing with problems where you need to know time series analysis, basic machine learning techniques, linear regressions, and causal inference. There are lots of kinds of data for which you'd need a more advanced statistics background to be able to analyze.

Figuring out the balance between computer science, statistics and math will really depend on the role you take, so these are just some of my general observations.

Why do you think so many people entering data science have Ph.D.s?

Data science is a new field now, and employers are looking for people with the qualifications to become a data scientist. Because it's such a new field, not that many people have much industry experience in this, so you have to find people who show some other signal that they'd be qualified for the position. Having a Ph.D. in a computational/quantitative background is a great choice usually, since they've already done plenty of research and data work. Ph.D. and Master's students with data experience often have qualities that are great for data science: learning quickly, asking questions, and being resilient.

Ph.D. and Master's students with data experience often have qualities that are great for data science: learning quickly, asking questions, and being resilient.

I think companies will start hiring more and more undergrads to fill data science roles in 5-10 years as there will be more people coming out with the right data science background. There are a ton more Sophomores at Harvard, for instance who want to become data scientists, then there were when I was a Sophomore. I think they view it as a promising and exciting career opportunity, of which I wholly agree.

Right now, there are plenty of MOOCs (Massive Open Online Courses) offering classes and certificates, and universities all over the world are offering their first data science class. For example, Harvard's first data science class and first predictive modeling class showed up in the 2013-2014 school year. These classes are perfect for undergrads who want to work on data.

If you're trying to hire data scientists and there are very few people with experience, those with Phds and Master's are good candidates. That will probably change in 5 to 10 years as there will be more undergraduates who come out with the right data science background.

Right now on Coursera, there's already a data science specialization, and at Harvard there's a new class called Data Science taught by Joe Blitzstein and Hanspeter Pfister. Joe is the same professor who taught the statistics class I loved.

In Spring 2014, a predictive modeling class started at Harvard. This is a class that focuses on Kaggle competitions. This kind of class is perfect for undergrads who want to work with data.

If you had to go back to when you were just starting out, what would you have focused on more, and what would you have focused on less?

I think my big regret in course selection in college was not taking programming classes my freshman year. Programming is so vital in data science — there's not that many roles for pure statisticians who don't code unless it's a giant company like Google or Amazon that might be specialized enough to need research statisticians. Programming is so essential that you can't get away with not doing it well.

When it comes to this term "data science", a lot of people are worried or claim there's a lot of hype around the field in that it's overblown. What's your take around this hype and craze around big data and data science?

It's definitely a bit overhyped right now, just like cloud computing and the mobile / local / social craze. However, just because it's overhyped doesn't mean it's not important. I think over the next few years, the hype will die down but the importance of data science will not.

Do you think that the need for data scientists will die down as tools get better?

Personally, I appreciate the new tools a lot. I think the job of the data scientist will change a lot over the next few years as the tool kit gets better.

We're always going to need people who can interpret results and distill insights into actionable plans to improve business.

However, I don't think the need for data scientists will decrease because we're always going to need people who can interpret results and distill insights into actionable plans to improve business. Data science is never going to run out of hard problems — there will always be the need for people to interpret results and communicate ideas. That's what I think data science is — it's distilling the data into actionable insights to improve product and business.

Tools will make what some data scientists do outdated, as some startups provide enterprise solutions and commoditize certain tasks. Even with the new enterprise tools, there will be a need for data scientists to be able to use the tools intelligently. You're going to want your data scientists to look at the results and think about how they can help the company directly.

How much domain expertise do you need in order to be a good data scientist? How much do you need to know about people's behaviors online, and does that drive the products to be built?

At Quora, I worked on a project that involved understanding user engagement. I was in a unique position while trying to understand that problem since I was an avid user of

Quora myself. When you have domain knowledge, you have an advantage in that you can make better hypotheses on what you're curious about before you even look at the data. You can then look at the data to gain a better intuition on why you were right or wrong. Domain expertise and the intuition that comes coupled with that can help a lot, especially if your models are complicated or you need to present them to an internal audience. The domain expertise facilitates the sharing of insightful stories that help explain the drivers of human behavior in your product. This is really different than some data sets on Kaggle where you aren't even given the column names (because of privacy) and don't really understand the data you're working with.

When you have domain knowledge, you have an advantage in that you can make better hypotheses on what you're curious about before you even look at the data.

You were choosing between quantitative finance and data science and eventually chose data science. Why did that happen, and what were the considerations when you were making that decision?

I think quantitative finance and data science are both really good options. I'm pretty sure that data science was the right option for me because I am just so excited to see how technology can change the way the world works and make everything work better. I felt like I wanted to be a part of that. I decided that in order for me to do this properly, I needed to be part of a consumer or enterprise technology firm where I was able to help make a product that empowered people to do things.

I also really like the teaching and communication aspects of data science — I found out that I enjoyed it when I got to help teach Statistics 110 at Harvard. Data Science has a lot more of this teaching and communication going on — often in quantitative finance all you need are your back-testing results.

I want to be some sort of evangelist for data, and convince people that data is useful. I feel that there's a lot more potential to do this in the tech sector. For tech, data is very new, while for finance, data is very old. I just found it exciting to be part of something where data was just getting a foothold. I wanted to be a part of something where technology is used to empower people and make the world better.