I'm very clear that I'm a big fan of Test Everything, which is any code change that you make, any feature that you introduce has to be in some experiment because, again, I've observed this sort of surprising result that even small bug fixes, even small changes can sometimes have surprising unexpected impact.
And so I don't think it's possible to experiment too much.
You have to allocate sometimes to these high-risk, high-reward ideas.
We're going to try something that's most likely to fail, but if it does win, it's going to be a home run.
And you have to be ready to understand and agree that most will fail.
And it's amazing how many times I've seen people come up with new designs or a radical new idea, and they believe in it, and that's okay.
I'm just cautioning them all the time to say, if you go for something big, try it out, but be ready to fail 80% of the time.
Welcome to Lenny's podcast, where I interview world-class product leaders and growth experts to learn from their hard-won experiences building and growing today's most successful products.
Today, my guest is Rani Kohavi.
Rani is seen by many as the world expert on A.B. testing and experimentation.
Just recently, he was VP and technical fellow of relevance at Airbnb, where he led their search experience team.
Prior to that, he was corporate vice president at Microsoft, where he led Microsoft's experimentation platform team.
Before that, he was director of data mining and personalization at Amazon.
He's currently a full-time advisor and instructor.
He's also the author of the go-to book on experimentation, called Trustworthy Online Controlled Experiments.
And in our show notes, you'll find a code to get a discount on taking his live cohort-based course on Maven.
In our conversation, we get super tactical about A.B. testing.
Rani shares his advice for when you should start considering running experiments at your company, how to change your company's culture to be more experiment-driven, what are signs your experiments are potentially invalid, why trust is the most important element of a successful experiment culture and platform, how to get started if you want to start running experiments at your company.
He also explains what actually is a P-value and something called Twyman's Law plus some hot takes about Airbnb and experiments in general.
This episode is for anyone who is interested in either creating an experiment-driven culture at their company or just fine-tuning one that already exists.
Enjoy this episode with Rani Kohavi after a short word from our sponsors.
This episode is brought to you by Mixpanel.
Take deep insights into what your users are doing at every stage of the funnel at a fair price that scales as you grow.
Mixpanel gives you quick answers about your users from awareness to acquisition through

retention.
And by capturing website activity, ad data, and multi-touch attribution right in Mixpanel, you can improve every aspect of the full user funnel.
Powered by first-party behavioral data instead of third-party cookies, Mixpanel is built to be more powerful and easier to use than Google Analytics.
Remember plans for teams of every size and see what Mixpanel can do for you at mixpanel.com slash friends slash Lenny.
And while you're at it, they're also hiring.
So check it out at mixpanel.com slash friends slash Lenny.
This episode is brought to you by Round.
Round is the private network built by tech leaders for tech leaders.
Round combines the best of coaching, learning, and authentic relationships to help you identify where you want to go and accelerate your path to get there, which is why their waitlist tops thousands of tech execs.
Round is on a mission to shape the future of technology and its impact on society.
Leading in tech is uniquely challenging, and doing it well is easiest when surrounded by leaders who understand your day-to-day experiences.
When we're meeting and building relationships with the right people, we're more likely to learn, find new opportunities, be dynamic in our thinking, and achieve our goals.
Learning and managing your network doesn't have to feel like networking.
Join Round to surround yourself with leaders from tech's most innovative companies.
Build relationships, be inspired, take action, visit round.tech slash apply, and use promo code Lenny to skip the waitlist.
That's round.tech slash apply.
Ronnie, welcome to the podcast.
Thank you for having me.
So you're known by many as maybe the leading expert on A-B testing and experimentation, which I think is something every product company eventually ends up trying to do, often badly.
And so I'm excited to dig quite deep into the world of experimentation and A-B testing to help people run better experiments.
So thank you again for being here.
Great goal.
Thank you.
Let me start with kind of a fun question.
What is maybe the most unexpected A-B test you've run, or maybe the most surprising result from an A-B test that you run?
Yeah.
So I think the opening example that I use in my book and in my class is the most surprising public example we can talk about.
And this was kind of an interesting experiment.
Somebody proposed to change the way that ads were displayed on Bing, the search engine.
And he basically said, let's take the second line and move it, promote it to the first line so that the title line becomes larger.

And when you think about that, and there's, you know, if you're going to look in my book or in the class, there's an actual diagram of what happened, the screenshots.
But if you think about it, it's just realistically, it looks like an idea.
Like, why would this be such a reasonable, interesting thing to do?
And indeed, when we went back to the backlog, it was on the backlog for months and languished there, and many things were rated higher.
But the point about this is it's trivial to implement.
So if you think about return on investment, we could get the data by having some engineer spend a couple of hours implementing it.
And that's exactly what happened.
Somebody, Ed Bing, who kept seeing this in the backlog and said, my God, we're spending too much time discussing it, I could just implement it.
And he did.
And he spent a couple of days implementing it, and as is, you know, the common thing at Bing, he launched the experiment.
And a funny thing happened.
We had an alarm, big escalation, something is wrong with the revenue metric.
Now this alarm fired several times in the past when there were real mistakes where somebody would log revenue twice or, you know, there's some data problem.
But in this case, there was no bug.
That simple idea increased revenue by about 12%.
And this is something that just doesn't happen.
We can talk later about Toyma's law, but that was the first reaction, which is this is too good to be true.
Let's find the bug.
And we did and we looked for several times and we replicated the experiment several times and there was nothing wrong with it.
This thing was worth $100 million at the time when Bing was a lot smaller.
And the key thing is it didn't hurt the user metric.
So it's very easy to increase revenue by doing theatrics that, you know, displaying more ads is a trivial way to raise revenue.
But it hurts the user experience and we've done the experiments to show that.
In this case, this was just a home run that improved revenue, didn't significantly hurt the guardrail metrics.
And so I was, we were like in awe of, you know, what a trivial change that was the biggest revenue impact of Bing in all its history.
And that was basically shifting in two lines, right?
Switching two lines in the search results, right?
And this was just moving the second line to the first line.
Now, you then go and run a lot of experiments to understand what happened here.
Is it the fact that the title line has a bigger font, sometimes different color?
So we ran a whole bunch of experiments and this is what usually happens.
We have a breakthrough.

You start to understand more about what can we do?
And there was suddenly a shift towards, okay, what are other things we could do that would allow us to improve revenue?
We came up with a lot of follow on ideas that helped a lot.
But to me, this was an example of a tiny change that was the best revenue generating idea in Bing's history and we didn't rate it properly, right?
Everybody gave this the priority that in hindsight it deserves and that's something that happens often.
I mean, we are often humbled by how bad we are at predicting the outcome of experiments.
This reminds me of a classic experiment at Airbnb while I was there and we'll talk about Airbnb in a bit.
The search team just ran a small experiment of what if we were to open a new tab every time someone clicked on a search result instead of just going straight to that listing and that was one of the biggest wins in search.
And by the way, I don't know if you know the history of this, but I tell about this in class.
We did this experiment way back around 2008, I think, and so this predates Airbnb.
And I remember it was heavily debated, like why would you open something in a new tab?
The users didn't ask for it.
There was a lot of pushback from the designers and we ran that experiment.
And again, it was one of these highly surprising results that made it that we learned so much from it.
So we first did this.
It was done in the UK for opening Hotmail and then we moved it to MSN so it would open Search in New Tab and all the set of experiments were highly, highly beneficial.
We published this and I have to tell you when I came to Airbnb, I talked to our joint friend Ricardo about this and it was sort of done.
It was very beneficial and then it was semi-forgotten, which is one of the things you learn about institutional memories.
When you have winners, make sure to address them and remember them.
So it was an Airbnb done for a long time before I joined that listings open in a new tab.
But other things that were designed in the future were not done and I reintroduced this to the team and we saw big improvements.
Shout out to Ricardo, our mutual friend who helped make this conversation happen.
There's this holy grail of experiments that I think people are always looking for of one hour of work and it creates this massive result.
I imagine this is very rare and don't expect this to happen.
I guess in your experience, how often do you find kind of one of these gold nuggets just lying around?
Yeah.
So again, this is a topic that's near and dear to my heart.
Everybody wants these amazing results and I show them in chapter one in my book.
Multiple of these small effort, huge gain, but as you said, they're very rare.

I think most of the time, the winnings are made sort of this inch by inch and there's a graph that I show in my book, a real graph of how Bing ads has managed to improve the revenue per thousand searches over time and every month you can see a small improvement and a small improvement.
Sometimes a degradation because of legal reasons or other things.
There was some concern that we were not marking the ads properly so you have to suddenly do something that you know is going to hurt revenue.
But yes, I think most results are inch by inch.
You improve small amounts, lots of them.
I think the best example that I can say is a couple of them that I can speak about.
One is that Bing, the relevance team, hundreds of people all working to improve Bing relevance.
They have a metric, we'll talk about, we see the overall evaluation criterion, but they have a metric that their goal is to improve it by 2% every year.
It's a small amount and that 2% you can see, here's a 0.1, here's a 0.15, here's a 0.2 and then they add up to around 2% every year, which is amazing.
Another example that I am allowed to speak about from Airbnb is the fact that we ran some 250 experiments in my tenure there in search relevance and again, small improvements added up.
This became overall a 6% improvement to revenue.
When you think about 6%, it's a big number, but it came out not of one idea, but many, many smaller ideas that each gave you a small gain.
In fact, again, there's another number I'm allowed to say of these experiments, 92% failed to improve the metric that we were trying to move.
Only 8% of our ideas actually were successful at moving the key metrics.
There's so many threads I want to follow here, but let me follow this one right here.
You just mentioned if 92% of experiments failed, is that typical in your experience seeing experiments run a lot of companies, what should people expect when they're running experiments?
What percentage should they expect to fail?
First of all, I published three different numbers for my career.
Overall at Microsoft, about 66%, two-thirds of ideas fail and don't think the 66% is accurate.
It's about two-thirds, and Bing, which is a much more optimized domain after we've been optimizing it for a while, the failure rate was around 85%.
It's harder to improve something you've been optimizing for a while.
Then at Airbnb, this 92% number is the highest failure rate that I've observed.
Now, I've quoted other sources that it's not that I worked at groups that were particularly bad, booking, Google ads, other companies published numbers that are around 80% to 90% failure rate of ideas.
This is where it's important, of experiments.
It is important to realize that when you have a platform, it's easy to get this number.
You look at how many experiments were run and how many of them launched.
Not every experiment maps to an idea.
It's possible that when you have an idea, your first implementation, you start an experiment,

boom, it's egregiously bad because you have a bug.
In fact, 10% of experiments tend to be aborted on the first date.
Those are usually not that the idea is bad, but that there is an implementation issue or something we haven't thought about that forces an abort.
You may iterate and pivot again, and ultimately, if you do two or three or four pivots or bug fixes, you may get to a successful launch, but those numbers of 80% to 92% failure rate are of experiments.
Very humbling.
I know that every group that starts to run experiments, they always start off by thinking that somehow they're different and their success rate is going to be much, much higher.
They're all humble.
You mentioned that you have this pattern of clicking a link and opening a new tab as a thing that just worked at a lot of different places.
Are there other versions of this?
Do you collect a list of here's things that often work when we want to move?
Yeah.
Are there some you could share?
I don't know if you have a list in your head.
I can give you two resources.
One of them is a paper that we wrote called Rules of Thumb.
What we tried to do at that time at Microsoft was to just look at thousands of experiments that run and extract some patterns.
It's one paper that we can then put in the notes.
But there's another more accurate, I would say, resource that's useful that I recommend to people.
It's a site called GoodUI.org.
GoodUI.org is exactly the site that tries to do what you're saying at scale.
The name is Jacob Linovsky.
He asks people to send them results of experiments, and he puts them into patterns.
There's probably 140 patterns, I think, at this point.
For each pattern, he says, well, who hasn't helped?
How many times and by how much?
You have an idea of this work three out of five times, and it was a huge win.
In fact, you can find that open in your window in there.
I feel like you feed that into chat, GPT, and you have basically a product manager creating a roadmap tool.
In general, by the way, this is all about a lot of that as institutional memory, which is can you document things well enough so that the organization remembers the successes and failures and learns from them?
I think one of the mistakes that some company makes is they launch a lot of experiments and never go back and summarize the learnings.
I've actually put a lot of effort in this idea of institutional learning of doing the quarterly meeting of the most surprising experiments.

By the way, surprising is another question that people often are not clear about.
What is a surprising experiment?
To me, a surprising experiment is one where the estimated result beforehand and the actual result differ by a lot, so that absolute value of the difference is large.
If you can expect something to be great and it's flat, well, you learn something, but if you expect something to be small and it turns out to be great, like that ad title promotion, then you've learned a lot.
Conversely, if you expect that something will be small and it's very negative, you can learn a lot by understanding why this was so negative, and that's interesting.
We focus not just on the winners, but also surprising losers, things that people thought would be a no-brainer to run, and then for some reason it was very negative.
Sometimes it's that negative that gives you insight.
Actually, I'm just coming up with one example of that that I should mention.
We were running this experiment at Microsoft to improve the Windows indexer, and the team was able to show on offline tests that it does much better at indexing, and they showed some relevance is higher and all these good things, and then they ran it as an experiment.
You know what happened?
Surprising result.
Indexing the relevance was actually high, but it killed the battery life.
Here's something that comes from left field that you didn't expect.
It was consuming a lot more CPU on laptops.
It was killing the laptops, and therefore, okay, we learned something.
Let's document it.
Let's remember this so that we now take this other factor into account as we design the next iteration.
What advice do you have for people to actually remember these surprises?
You said that a lot of it is institutional.
What do you recommend people do so that they can actually remember this when people leave, say, three years later?
Document it.
We had a large deck internally of these successes and failures, and we encourage people to look at them.
The other thing that's very beneficial is just to have your whole history of experiments and do some ability to search by keywords.
I have an idea, type a few keywords and see if from the thousands of experiments that ran.
By the way, these are very reasonable numbers.
At Microsoft, just to let you know, when I left in 2019, we were on a rate of about 20,000 to 25,000 experiments every year.
Every working day, we were starting something like 100 new treatments, big numbers.
When you're running in a group like Bing, which is running thousands and thousands of experiments, you want to be able to ask, has anybody did an experiment on this, or this, or this, and so that searching capability is in the platform?

More than that, I think just doing the quarterly meeting of the most interesting, sorry, not just successful, most interesting experiments is very key.
That also helps a flywheel of experimentation.
This is a good segue to something I wanted to touch on, which is there's often a, I guess, a weariness of running too many experiments and being too data-driven, and the sense that experimentation just leads you to these micro-optimizations, and you don't really innovate and do big things.
What's your perspective on that?
Can you be too experiment-driven in your experience?
I'm very clear that I'm a big fan of test everything, which is any code change that you make, any feature that you introduce has to be in some experiment because, again, I've observed this sort of surprising result that even small bug fixes, even small changes can sometimes have surprising unexpected impact.
I don't think it's possible to experiment too much.
I think it is possible to focus on incremental changes because some people say, well, if we only tested 17 things around this and you have to think about it's not just, it's like in stock, you need a portfolio, you need some experiments that are incremental that move you in the direction that you know you're going to be successful over time if you just try enough, but some experiments, you have to allocate sometimes to these high-risk, high-reward ideas.
We're going to try something that's most likely to fail, but if it does win, it's going to be a home run, and so you have to allocate some efforts to that, and you have to be ready to understand and agree that most will fail.
Most of these, and it's amazing how many times I've seen people come up with new designs or a radical new idea, and they believe in it, and that's okay.
I'm just cautioning them all the time to say, if you go for something big, try it out, but be ready to fail 80% of the time.
One true example that, again, I'm able to talk about because we put it in my book is we were at Bing trying to change the landscape of search, and one of the ideas, the big ideas, was we're going to integrate with social, so we hooked into the Twitter, Firehose Feed, and we hooked into Facebook, and we spent 100% years on this idea, and it failed.
You don't see it anymore.
It existed for about a year and a half, and all the experiments were just negative to flat, and it was an attempt.
It was fair to try it.
I think it took us a little long to fail to decide that this is a failure, but at least we had the data.
We had hundreds of experiments that we tried.
None of them were a breakthrough, and I remember sort of mailing Chi Lu with some statistics showing that it's time to abort.
It's time to fail on this.
And he decided to continue more, and it's a million-dollar question, you know, do you continue, and then maybe the breakthrough will come next month,

or do you abort?
And a few months later, we abort it.
That reminds me of, at Netflix, they tried a social component that also failed at Airbnb.
Early on, there was a big social attempt to make, like, here's your friends have stayed at these AirBBs completely, not had no impact.
So maybe that's one of these learnings that we should document.
Yeah, this is hard.
This is hard.
And, but that's, again, that's the value of experiments, which are this oracle that gives you the data.
You may be excited about things.
You may believe it's a good idea, but ultimately, the oracle is the controlled experiment.
It tells you whether users are actually benefiting from it, whether you and the users, the company, and the user.
There's obviously a bit of overhead and downsides running an experiment, setting all up and making sure, you know, analyzing the results.
Is there anything that you ever don't think is worth AB testing?
First of all, there are some necessary ingredients to AB testing.
And I'll just say, all right, not every domain is amenable to AB testing, right?
You can't AB test mergers and acquisitions, right?
That's something that happens once you either acquire or you don't acquire.
So you do have to have some necessary ingredient.
You need to have enough units, mostly users, in order for the statistics to work out.
So yeah, if you're too small, then maybe too early to AB test.
But what I find is that in software, it is so easy to run AB testing and it is so easy to build a platform.
I don't say it's easy to build a platform, but once you build the platform, the incremental cost of running an experiment should approach zero.
And we got to that at Microsoft, where after a while, the cost of running experiments was so low that nobody was questioning the idea that everything should be experimented with.
Now, I don't think we were there at Airbnb, for example.
The platform at Airbnb was much less mature and required a lot more analysts in order to interpret the results and to find issues with it.
So I do think there's this trade-off.
You're willing to invest in the platform.
It is possible to get the marginal cost to be close to zero.
But when you're not there, it's still expensive.
And there may be reasons why not to run AB tests.
You talked about how you may be too small to run AB tests.
And this is a constant question for startups, is when should we start running AB tests?
Do you have kind of a heuristic or a real thumb of just like, here's a time you should really start thinking about running.
Yeah, yeah, a million-dollar question that everybody asked.

So I actually will put this in the notes, but I gave a talk last year,
what I call it is practical defaults.
And one of the things I show there is that unless you have at least tens of thousands of users,
the math, the statistics just don't work out for most of the metrics that you're interested in.
In fact, I gave an actual practical number of a retail site with some conversion rate,
trying to detect changes that are at least 5% beneficial,
which is something that startups should focus on.
They shouldn't focus on the 1%.
They should focus on the 5% and 10%.
Then you need something like 200,000 users.
So start experimenting when you're in the tens of thousands of users.
You'll be only able to detect large effects.
And then once you get to 200,000 users, then the magic starts happening.
Then you can start testing a lot more.
Then you have the ability to test everything and make sure that you're not degrading
and getting value out of experimentation.
So you ask for rule of thumb, 200,000 users, you're magical.
Below that, start building the culture.
Start building the platform.
Start integrating so that as you scale, you start to see the value.
Love it.
Coming back to this kind of concern people have of experimentation,
keeps you from innovating and taking big bets.
I know you have this framework, Overall Evaluation Criterion.
And I think that helps with this.
Can you talk a bit about that?
The OEC, or the Overall Evaluation Criterion,
is something that I think many people that start to dabble and maybe testing miss.
And the question is, what are you optimizing for?
And it's a much harder question that people think because it's very easy to say,
we're going to optimize for money, revenue.
But that's the wrong question because you can do a lot of bad things that will improve revenue.
So there has to be some countervailing metric that tells you,
how do I improve revenue without hurting the user experience?
Okay, so let's take a good example with search.
You can put more ads on the page and you will make more money.
There's no doubt about it.
You will make more money in the short term.
The question is, what happens to the user experience
and how is that going to impact you in the long term?
So we've run those experiments and we were able to map out,
this number of ads causes this much increase to churn.
This number of ads causes this much increase to the time that users take to find a successful result.

And we came up with an OEC that is based on these metrics
that allows you to say, okay, I'm willing to take this additional money
if I'm not hurting the user experience by more than this much.
So there's a trade off there.
One of the nice ways to phrase this is a constraint optimization problem.
I want you to increase revenue, but I'm going to give you
a fixed amount of average real estate that you can use.
So for one query, you can have zero ads.
For another query, you can have three ads.
For a third query, you can have wider, bigger ads.
I'm just going to count the pixels that you take, the vertical pixels.
And I will give you some budget.
And if you can, under the same budget, make more money, you're good to go.
Right?
So that, to me, turns the problem from a badly defined,
let's just make more money, right?
Any page can start plastering more ads and make more money short term,
but that's not the goal.
The goal is long term growth and revenue.
Then you need to insert these other criteria.
And what am I doing to the user experience?
One way around it is to put this constraint.
Another one is just to have these other metrics.
Again, something that we did to look at the user experience.
How long does it take the user to reach a successful click?
What percentage of sessions are successful?
These are key metrics that were part of the overall evaluation criterion that we've used.
I can give you another example.
By the way, from the hotel industry or Airbnb that we both worked at,
you can say I want to improve conversion rate,
but you can be smarter about it and say,
it's not just enough to convert a user to buy or to pay for a listing.
I want them to be happy with it several months down the road when they actually stay there.
Right?
So that could be part of your OEC to say,
what is the rating that they will give to that listing when they actually stay there?
And that causes an interesting problem because you don't have this data now.
You're going to have it three months from now when they actually stay.
So you have to build the training set that allows you to make a prediction
about whether this user, whether Lenny is going to be happy at this cheap place
or whether no, I should offer him something more expensive
because Lenny likes to stay at nicer places where the water actually is hot
and comes out of the faucet.

That is true.
Okay, so it sounds like the core to this approach is basically have a kind of a drag metric that makes sure you're not hurting something that's really important to the business and then being very clear on what's the long-term metric we care most about.
To me, the key here, the key word is lifetime value, which is you have to define the OEC such that it is causally predictive of the lifetime value of the user.
Right?
And that's what causes you to think about things properly.
Am I doing something that just helps me short-term or am I doing something that will help me the longer?
Once you put that model of lifetime value, people say, okay, what about retention rates?
You can measure that.
What about the time to achieve a task?
We can measure that.
And those are these countervailing metrics that make the OEC useful.
And to understand these longer-term metrics, what I'm hearing is use kind of models and forecasts and predictions.
Or would you suggest sometimes use like a long-term holdout or some other approach?
Like, what do you find is the best way to see these long-term?
So there's two ways that I like to think about it.
One is you can run long-term experiments for the goal of learning something.
So I mentioned that at Bing, we did run these experiments where we increase the ads and decrease the ads so that we will understand what happens to key metrics.
The other thing is you can just build models that use some of our background knowledge or use some data science to look historical.
I'll give you another good example of this.
When I came to Amazon, one of the teams that reported to me was the email team that it was not the transactional emails when you buy something, you get an email, but it was the team that sent these recommendations.
Here's a book by an author that you bought.
Here's a product that we recommend.
And the question is, how do we give credit to that team?
And the initial version was, well, whenever a user comes from the email and purchases something on Amazon, we're going to give that email credit.
Well, it turned out this had no counter-availability metric.
The more emails you send, the more money you're going to credit the team.
And so that led to spam.
Literally a really interesting problem.
The team just ramped up the number of emails that they were sending out and claimed to make more money and their fitness function improved.
And then so then we backed up and then we said, okay, there we can either phrase this as a constraint satisfaction problem.

You're allowed to send user an email every X days.
Or, which is what we ended up doing is, let's model the cost of spamming the users.
Okay, what's that cost?
Well, when they unsubscribe, we can't mail them.
Okay, so we did some data science study on the side and we said,
what is the value that we're losing from an unsubscribe?
Right?
And we came up with a number.
It was a few dollars.
But the point was, now we have this counter-availability metric.
We say, here's the money that we generate from the emails.
Here's the money that we're losing a long-term value.
What's the trade-off?
And then when we started to incorporate this formula,
more than half the campaigns that were being sent were negative.
Okay, so it was a huge insight at Amazon about how to send the right campaigns.
And this led, and this is what I like about these discoveries,
this fact that we integrated the unsubscribe led us to a new feature
to say, well, let's not lose their future lifetime value through email.
When they unsubscribe, let's offer them by default to unsubscribe from this campaign.
So when you get an email, there's a new book by the author,
the default to unsubscribe would be unsubscribe me from author emails.
And so now the negative, the counter-availing metric is much smaller.
And so again, this was a breakthrough in our ability to send more emails
and understand based on what users were unsubscribing from,
which ones are really beneficial.
I love the surprising results.
We all love them.
I mean, this is the humbling reality, and people talk about the fact that
AB testing sometimes leads you to incremental.
I actually think that many of these small insights lead to fundamental insights about
you know, which areas to go, some strategies we should take,
some things we should develop helps a lot.
This makes me think about how every time I've done a full redesign of a product,
I don't think ever has it ever been a positive result.
And then the team always ends up having to claw back what they just
hurt and try to figure out what they messed up.
Is that your experience too?
Absolutely.
Yeah, in fact, I've published some of these in LinkedIn posts showing
a large set of big launches that redesigns and dramatically failed.
And it happens very often.
So the right way to do this is to say, yes, we want to do a redesign,

but let's do it in steps and test on the way and adjust.
So you don't need to take 17 new changes that many of them are going to fail.
Start to move incrementally in a direction that you believe is beneficial
and just on the way.
The worst part of those experiences I find is it took like, I don't know,
three, six months, three or six months to build it.
And by the time it's launched, it's just like, we're not going to
un-launch this.
Everyone's been working in this direction.
All the new features are assuming this is going to work.
And you're basically stuck, right?
I mean, this is a sunk cost fallacy, right?
We invested so many years in it.
Let's launch this even though it's bad for the user.
No, that's terrible.
Yeah, yeah.
So this is the other advantage of recognizing this humble reality
that most ideas fail, right?
If you believe in that statistics that I published,
then doing 17 changes together is more likely to be negative.
Do them in smaller increments.
Learn from, it's called OFAT, one factor at a time.
Do one factor, learn from it and adjust.
Of the 17, maybe you have four good ideas.
Those are the ones that will launch and be positive.
I generally agree with that and always try to avoid a big redesign,
but it's hard to avoid them completely.
There's often team members that are really passionate
and like we just need to rethink this whole experience.
We're not going to incrementally get there.
Have you found anything effective in helping people either see this
perspective or just making a larger bet more successful?
By the way, I'm not opposed to large redesigns.
I try to give the team the data to say, look,
here are lots of examples where big redesigns fail.
Try to decompose your redesign.
If you can't decompose the one factor at a time,
do a small set of factors at a time and learn from these smaller changes
what works and what doesn't.
Now, it's also possible to do a complete redesign.
I'm just, as you said yourself, they'd be ready to fail, right?
I mean, do you really want to work on something for six months or a year
and then run the AAB test and realize that you've hurt revenues

or other key metrics by several percentage points
and a data-driven organization will not allow you to launch?
What are you going to write in your annual review?
Yeah, but nobody ever thinks it's going to fail.
I think, no, we got this.
We've talked to so many people.
But I think organizations that start to run experiments
are humbled early on from the smaller changes.
Yeah.
Right?
You're right.
Nobody, I'll tell you a funny story.
When I came from Amazon to Microsoft, I joined the group.
And for one reason or another, that group disbanded a month after I joined.
And so people came to me and said,
look, you just joined the company.
You're at partner level.
You figure out how you can help Microsoft.
And I said, I'm going to build an experimentation platform
because nobody at Microsoft is running experiments.
And more than 50% of ideas in Amazon that we tried failed.
And the classical response was, we have better PMs here.
Right?
There was this complete denial that it's possible
that 50% of ideas that Microsoft is implementing
in a three-year development cycle, by the way.
This is how long it took Office to release.
It was a classical.
Every three years we release.
And the data came about showing that Bing was the first
to truly implement experimentation at scale.
And we shared with the rest of the companies the surprising results.
And so when Office was, and this was credit to Chi Lu and Satya Nadella,
they were ones that says, Ronnie, you try to get Office to run experiments.
We'll give you the air support.
And it was hard, but we did it.
It took a while, but Office started to run experiments.
And they realized that many of their ideas were failing.
You said that there's a site of failed redesigns.
Was that in your book, or is that a site that you can point people to
to kind of help build this case?
I teach this in my class, but I think I've posted this on LinkedIn
and answered some questions.

I'm happy to put that in the notes.
Okay, cool.
We'll put that in the show notes because I think that's the kind of data
that often helps convince a team.
Maybe we shouldn't rethink this entire onboarding flow from scratch.
Maybe we should kind of iterate towards and learn as we go.
This episode is brought to you by Epo.
Epo is a next generation, A.B. testing platform built by Airbnb alums
for modern growth teams.
Companies like DraftKings, Zapier, ClickUp, Twitch, and Cameo
rely on Epo to power their experiments.
Wherever you work, running experiments is increasingly essential.
But there are no commercial tools that integrate with a modern growth team stack.
This leads to waste of time building internal tools
or trying to run your own experiments through a clunky marketing tool.
When I was at Airbnb, one of the things that I loved most about working there
was our experimentation platform where I was able to slice and dice data
by device types, country, user stage.
Epo does all that and more delivering results quickly,
avoiding annoying prolonged analytic cycles,
and helping you easily get to the root cause of any issue you discover.
Epo lets you go beyond basic click-through metrics
and instead use your NorthStar metrics like activation, retention, subscription, and payments.
Epo supports tests on the front end, on the back end, email marketing,
even machine learning clients.
Check out Epo at geteppo.com.
That's getepo.com and 10x your experiment velocity.
I like this 80% of the time you will fail, so be ready for that, right?
What people usually expect is, my redesign is going to work.
No, you're most likely going to fail, but if you do succeed, it's a breakthrough.
I like this 80% of the time you're going to fail,
but it's not going to work.
It's going to be a breakthrough.
It's going to be a breakthrough.
It's going to be a breakthrough.
It's going to be a breakthrough.
It's going to be a breakthrough.
I like this 80% rule of thumb.
Is that just a simple way of thinking about it?
That's my rule of thumb.
I've heard people say it's 70% or 80%, but it's in that area where I think,
when you talk about how much to invest in the known versus the high-risk, high-reward,
that's usually the right percentage that most organizations end up doing this allocation.

You interviewed Shreyas.
I think he mentioned that at Google, it's like 70%.
The search and ads, and it's a 20% for some of the apps and new stuff,
and then it's the 10% for infrastructure.
Yeah.
I think the most important point there is, if you're not running an experiment,
70% of stuff you're shipping is hurting your business.
Well, it's not hurting, it's flat to negative.
Some of them are flat.
By the way, flat to me, if something is not that sick, that's a no-ship,
because you've just introduced more code.
There is a maintenance overhead to shipping your stuff.
I've heard people say, look, we already spent all this time.
The team will be demotivated if we don't ship it, and I'm like, no, that's wrong, guys.
Let's make sure that we understand that shipping this project that has no value
is complicating the code base.
Maintenance costs will go up.
You don't ship on flat unless it's a legal requirement.
When legal comes along and says you have to do X or Y, you have to ship on flat or even negative,
and that's understandable.
But again, I think that's something that a lot of people make the mistake of saying,
legal told us we have to do this.
Therefore, we're going to take the hits.
No, legal gave you a framework that you have to work under, try three different things,
and ship the one that hurts the least.
I love that.
It reminds me when Airbnb launched the rebrand,
even that they ran as an experiment with the entire homepage redesign,
the new logo, and all that.
I think there was a long-term holdout even,
and I think it was positive in the end, from what I remember.
Speaking of Airbnb, I want to chat about Airbnb briefly.
I know you're limited in what you can share,
but it's interesting that Airbnb seems to be moving in this other direction
where it's becoming a lot more top-down, Brian vision-oriented,
and Brian's even talked about how he's less motivated to an experiment.
He doesn't want to run as many experiments as they used to.
Things are going well, and so it's hard to argue with the success potentially.
You work there for many years.
You ran the search team, essentially.
I guess just what was your experience like there,
and then roughly what's your sense of how things are going, where it's going?
As you know, and I'm restricted from talking about Airbnb,

I will say a few things that I am allowed to say.
One is, in my team, in search relevance, everything was A.B. tested.
While Brian can focus on some of the design aspects, the people who are actually doing
the neural networks and the search, everything was A.B. tested to hell.
Nothing was launching without an A.B. test.
We had targets around improving certain metrics, and everything was that A.B. test.
Other teams, some did, some did not.
I will say that when you say things are going well, I think we don't know the counterfactual.
I believe that had Airbnb kept people like Greg Greeley,
which was pushing for a lot more data-driven, and had Airbnb run more experiments,
it would have been in a better state than today.
But it's the counterfactual we don't know.
That's a really interesting perspective.
Airbnb is such an interesting natural experiment of a way of doing things differently.
There's de-emphasizing experiments.
And also, they turned off paid ads during COVID.
And I think, I don't know where it is now, but it feels like it's become a much
smaller part of the growth strategy.
Who knows if they've ramped it up to back to where it is today.
But I think it's going to be a really interesting case study looking back,
I don't know, five, 10 years from now.
It's a one-off experiment where it's hard to assign value to some of the things that Airbnb is doing.
I personally believe it could have been a lot bigger,
and a lot more successful if it had run more controlled experiments.
But I can't speak about some of those that I ran and that showed
that some of the things that were initially untested were actually negative and could be better.
All right. Mysterious.
One more question, Airbnb, you were there during COVID, which was quite a wild time for Airbnb.
We had sunshine on the podcast talking about all the craziness that went on when travel
basically stopped.
And there was a sense that Airbnb was done and travel is not going to happen for years and years.
What's your take on experimentation in that world where you have to really move fast,
make crazy decisions, make big decisions?
What did you, what was like during that time?
So I think actually in a state like that, it's even more important to run AB tests,
because what you want to be able to see is if we're making this change,
is it actually helping in the current environment?
There's this idea of external generalizability.
Is it going to work out now during COVID?
Is it going to generalize later on?
These are things that you can really answer with the controlled experiments.
And sometimes it means that you might have to replicate them six months down
when COVID, say, is not as impactful as it is.

Saying that you have to make decisions quickly, to me, I'll point you to the success rate.
Like if in peacetime, you're wrong two thirds to 80% of the time,
why would you be suddenly right in wartime, right in COVID time?
So I don't believe in the idea that because bookings went down materially,
the company should certainly not be data driven and do things differently.
I think if everybody stayed the course, did nothing,
the revenue would have gone up in the same way.
Fascinating.
In fact, if you look at one investment, one big investment that was done at the time
was online experiences.
And the initial data wasn't very promising, and I think today it's a footnote.
Yeah, another case study for the history books, Airbnb experiences.
I want to shift a little bit and talk about your book, which you mentioned a couple of times.
It's called Trustworthy Online Controlled Experiments.
And I think it's basically the book on A-B testing.
Let me ask you just what surprised you most about writing this book and putting it out
and in the reaction to it?
I was pleasantly surprised that it sold more than what Cambridge predicted.
So when first we were approached by Cambridge after a tutorial that we did to write a book,
I was like, I don't know.
This is too small of an inch area.
And they were saying, so you'll be able to sell a few thousand copies and help the world.
And I found my co-authors, which are great.
And we wrote a book that we thought is not statistically oriented, has fewer formulas
than you normally see, and focuses on the practical aspects and on trust, which is the key.
The book, as I said, was more successful.
It sold over 20,000 copies in English.
It was translated to Chinese, Korean, Japanese, and Russian.
And so it's great to see that we helped the world become more data driven with experimentation.
And I'm happy because of that.
And I was pleasantly surprised.
By the way, all proceeds from the book are donated to the charity.
So if I'm pitching the book here, there is no financial gain for me from having more copies
sold.
I think we made that decision, which was a good decision.
All proceeds go with the charity.
Amazing.
I didn't know that.
We'll link to the book in the show notes.
You talked about how trust is in the title.
You just mentioned how important trust is to experimentation.
A lot of people talk about how do I run experiments faster?
You focus a lot on trust.

Why is trust so important in running experiments?
So to me, the experimentation platform is the safety net.
And it's an oracle.
So it serves really two purposes.
The safety net means that if you launch something bad, you should be able to abort quickly.
Right?
Safe deployments, safe velocity.
There are some names for this.
But this is one key value that the platform can give you.
The other one, which is the more standard one, is at the end of the two-week experiment,
we will tell you what happened to your key metric and do many of the other surrogate
and debugging and guardrail metrics.
Trust builds up.
It's easy to lose.
And so to me, it is very important that when you present this and say,
this is science, this is a controlled experiment, this is the resolve,
you better believe that this is trustworthy.
And so I focus on that a lot.
I think it allowed us to gain the organizational trust that this is really.
And the nice thing is when we built all these checks to make sure that the experiment is correct,
if there was something wrong with it, we would stop and say,
hey, something is wrong with the experiment.
And I think that's something that some of the early implementations in other places
did not do.
And it was a big mistake.
I've mentioned this in my book, so I can mention this here.
Optimizedly, in its early days, we're very statistically naive.
They sort of said, hey, we're real time.
We can compute your p-values in real time.
And then you can stop an experiment when the p-value is statistically significant.
That is a big mistake that inflates what's called type one error or the false positive rate
materially.
So if you think you've got a 5% type one error or you aim for that p-value less than 0.05,
using real time sort of p-value monitoring to optimize the offered,
you would probably have a 30% error rate.
So what this led is that people that started using Optimizedly
thought that the platform was telling them they're very successful.
But when they actually started to see, well, I told us this is positive revenue,
but I don't see this over time, like by now we should have made double the money.
So their questions started to come up around the trust in the platform.
There's a very famous post that somebody wrote about how Optimizedly almost got me fired
by a person who basically said, look, I came to the organ.
I said, we have all these successes, but then I said something is wrong.

And he tells of how he ran an AA test when there is no difference between the A and the B
and Optimizedly told them that it was statistically significant too many times.
Optimizedly learned.
Optimizedly, several people pointed, I pointed this out in my Amazon review of the book
that the Optimizedly authors wrote early on.
I said, hey, you're not doing the statistics correctly.
Ramesh Johari at Stanford pointed this out, became a consultant to the company,
and then they fixed it.
But to me, that's a very good example of how to lose trust.
They lost a lot of trust in the market.
They lost all this trust because they built something that had very much inflated erroring.
That is pretty scary to think about.
You've been running all these experiments and they weren't actually telling you accurate results.
What are signs that what you're doing may not be valid if you're starting to run experiments?
And then just how do you avoid having that situation?
What kind of tips can you share for people trying to run experiments?
There's a whole chapter of that in my book, but I'll say maybe one of the things that
is the most common occurs by far, which is a sample ratio mismatch.
Now, what is a sample ratio mismatch?
If you design the experiment to send 50% of users to control and 50% of users to treatment,
it's supposed to be a random number or hash function.
If you get something off from 50%, it's a red flag.
So let's take a real example.
Let's say you're running an experiment and it's large, it's got a million users,
and you got 50.2.
So people say, well, I don't know.
It's not going to be exactly the same as 50.2 reasonable or not.
Well, there's a formula that you can plug in.
I have a spreadsheet available for those that are interested.
And you can tell here's how many users are in control.
Here's how many users are having treatment.
My design was 50.50.
And it tells you the probability that this could have happened by chance.
Now, in a case like this, you plug in the numbers.
It might tell you that this should happen one in half a million experiments.
Well, unless you've run half a million experiments,
very unlikely that you would get a 50.2 versus 49.8 split.
And therefore, something is wrong with the experiment.
Now, people, I remember when we first implemented this check,
we were surprised to see how many experiments suffered from this.
Right?
And there's a paper that was published in 2018
and where we share that Ed and Microsoft,

even though we'd be running experiments for a while,
is around 8% of experiments that suffered from the sample regime as much.
And it's a big number.
Now, think about this.
You're running 20,000 experiments a year.
So many of them, 8% of them are invalid.
And somebody has to go down and understand what happened here.
We know that we can't trust the results, but why?
So over time, you begin to understand there's something wrong with the data pipeline.
There's something that happens with bots.
Bots are a very common factor for causing a sample ratio mismatch.
So there's a whole, that paper that was published by my team,
talks about how to diagnose sample ratio mismatches.
In the last, about a year and a half,
it was amazing to see all these third-party companies implement sample ratio mismatches.
And all of them were reporting, oh my God, 6%, 8%, 10%.
So yeah, they were, it's sometimes fun to go back and say,
how many of your results were in the past were invalid
before you had this sample ratio mismatch test?
That is frightening.
Is the most common reason this happens is you're assigning users
in the wrong place in your code?
So when you say most common, I think the most common is bots.
Somehow they hit the controller, the treatment in different proportions.
Because you changed the website, the bot may fail to parse the page
and try to hit it more often.
That's a classical example.
Another one is just the data pipeline.
We've had cases where we were trying to remove bad traffic under certain conditions
and it was skewed because of the control and treatment.
I've seen people that started an experiment in the middle of the site on some page,
but they don't realize that some campaign is pushing people from the side.
So there's multiple reasons.
It is surprising how often this happens.
And I'll tell you a funny story, which is when we first added this test to the platform,
we just put a banner saying, you have a sample ratio mismatch, do not trust these results.
And we noticed that people ignored it.
They were starting to present results that had this banner.
And so we blanked out the scorecard.
We put a big red, can't see this result.
You have a sample ratio mismatch.
Click OK to expose the results.
And why do we need that OK?

We need that OK button because you want to be able to debug the reasons.
And sometimes the metrics help you understand why you have a sample ratio mismatch.
So we blanked out the scorecard.
We had this button and then we started to see that people pressed the button
and still presented the results of experiments with sample ratio mismatch.
So we ended up with an amazing compromise,
which is every number in the scorecard was highlighted with a red line.
So that if you took a screenshot, other people could tell you had a sample ratio mismatch.
Freaking product managers.
This is intuition.
People just say, oh, my Instagram was small.
Therefore, I can still present the results.
People want to see success.
I mean, this is a natural bias.
And then we have to be very conscientious and fight that bias
and say when something looks too good to be true, investigate.
Which is a great segue to something you mentioned briefly,
something called Toyman's Law.
Yeah.
Can you talk about that?
Yeah.
So Toyman's Law, the general statement is if any figure that looks
interesting or different is usually wrong.
It was first said by this person in the UK who worked on radio media,
but I'm a big fan of it.
And my main claim to people is if the result looks too good to be true,
if you suddenly moved your normal movement of an experiment is under 1%
and you suddenly have a 10% movement, hold the celebratory dinner.
Like, it was just your first reaction, right?
Let's take everybody to a fancy dinner because we just improved revenue
by millions of dollars.
Hold that dinner, investigate.
See, because there's a large probability that something is wrong with the result.
And I will say that 9 out of 10 when we call out Toyman's Law,
it is the case that we find some flaw in the experiment.
Now, there are obviously outliers, right?
That first experiment that I showed where we promoted the made long ad titles
that was successful, but that was replicated multiple times
and double and triple checked and everything was good about it.
Many other results that were so big turn out to be false.
So I'm a big, I'm a big, big fan of Toyman's Law.
There's a deck I can also give this in a note where I shared some real examples of Toyman's Law.
Amazing.

I want to talk about rolling this out of companies and things that run,
you run into that fail.
But before I get to that, I'd love for you to explain just p-value.
I know that people kind of misunderstand it.
And this might be a good time just to help people understand.
What is it actually telling you at p-value of say 0.05?
I don't know if this is the right forum for explaining p-values
because the definition of a p-value is simple.
What it hides is very complicated.
So I'll say one thing which is many people assign one minus p-value
as the probability that your treatment is better than control.
So you ran an experiment, you got a p-value of 0.02.
They think there's a 98% probability that the treatment is better than the control.
That is wrong.
Okay, so rather than defining p-values, I want to caution everybody
that the most common interpretation is incorrect.
p-value assumes, it's a conditional probability or an assumed probability.
It assumes that the null hypothesis is true.
And we're computing the probability that the data we're seeing
matches the hypothesis.
That's null hypothesis.
In order to get the probability that most people want,
we need to apply Bayes' rule and invert the probability
from the probability of the data given the hypothesis
to the probability that the hypothesis is given the data.
For that, we need an additional number which is the probability,
the prior probability that the hypothesis that you're testing
is successful or not.
That's an unknown.
What we do is we can take historical data and say,
look, people fail two-thirds of the time or 80% of the time.
And we can apply that number and compute that.
We've done that in a paper that I will give in the notes
so that you can assess the number that you really want
that what's called a false positive risk.
So I think that's something for people to internalize
that what you really want to look at is this false positive risk
which tends to be much, much higher than the 5% that people think.
So if you're, I think the classical example in the Airbnb
where the failure rate was very, very high
is that when you get a statistically significant resolve,
let me actually hold a note so that I know how to have the actual number.
If you're at Airbnb where the success rate of,

or Airbnb search where the success rate is only 8%,
if you get a statistically significant result
with a p-value less than 0.05, there is a 26% chance
that this is a false positive result, right?
It's not 5%, it's 26%.
So that's the number that you should have in your mind.
And that's why when I worked at Airbnb,
one of the things we did is we said, okay,
if you're less than 0.05 but above 0.01, rerun, replicate.
When you replicate, you can combine the two experiments
and get a combined p-value using something called
Fisher's Method or Stauffer's Method.
And that gives you the joint probability
and that's usually much, much lower.
So if you get 2.05 or something like that,
then the probability that you've got them is much, much lower.
Wow, I have never heard it described that way.
It makes me think about how even data scientists and our teams
are always just like, this isn't perfect,
like we're not 100% sure this experiment is positive,
but on balance, if we're launching positive experiments,
we're probably doing good things.
It's okay if sometimes we're wrong.
By the way, it's true on balance,
you're probably better than 50-50,
but people don't appreciate how much that 26% that I mentioned is high.
And the reason that I want to be sure
is that I think it leads to this idea of the learning,
the institutional knowledge,
which you want to be able to say is,
share with the audience success.
And so you want to be really sure that you're successful.
So by lowering the p-value, by forcing teams
to work with the p-value, maybe below 0.01,
and do replication on hires,
then you can be much more successful.
And the false positive rate will be much, much lower.
Fascinating, and also shows the value of keeping track
of just what percentage experiments
are failing historically at the company
or within that specific product.
Say someone listening wants to start running experiments,
say they have tens of thousands of users at this point,

what would be the first couple of steps you'd recommend?
Well, so if they have somebody in the org
that has previously been involved experiment,
that's a good way to consult internally.
I think that the key decision is whether you want to build or buy.
There's a whole series of eight sessions
that I posted on LinkedIn
where I invited guest speakers to talk about those problems.
So if people are interested,
they can look at how what the vendors say,
and what agencies said about build versus buy question.
And it's usually not a zero one.
It's usually a both.
You build some and you buy some,
and it's a question of do you build 10% or do you build a 90%?
I think for people starting,
the third-party products that are available today are pretty good.
This wasn't the case when I started working.
So when I started building or running experiments at Amazon,
we were building the platform because nothing existed.
Same at Microsoft.
I think today there's enough vendors
that provide good experimentation platforms
that are trustworthy that I would say
not a good way to consider using one of those.
So you're at a company where there's resistance
to experimentation and A.B. testing,
whether it's a startup or a bigger company,
what have you found works in helping shift that culture
and how long does that usually take,
especially at a larger company?
My general experience is with Microsoft,
where we went with this beachhead of Bing.
We were running a few experiments,
and then we were asked to focus on Bing,
and we scaled experimentation
and built a platform at Scale at Bing.
Once Bing was successful,
and we were able to share all these surprising results,
I think many, many more people in the company were amenable.
And it was also the case that helped a lot,
that there's the usual cross-pollination people
from Bing move out to other groups,

and that helped these other groups say,
hey, there's a better way to build software.
So I think if you're starting out,
find a place, find a team where experimentation
is easier to run, and by that I mean,
they're launching often.
Don't go with a team that launches every six months,
or office used to launch every three years.
Go with a team that launches frequently.
They're running on sprints, they launch every week or two,
sometimes they launch daily.
I mean, Bing used to launch multiple times a day.
And then make sure that you understand the question of the OEC.
Is it clear what they're optimizing for?
Right, there are some groups where you can come up with a good OEC.
Some groups are harder.
You know, I remember one funny example was the Microsoft.com website,
which this is not MSN, this is Microsoft.com,
has like multiple different constituencies
that are trying to determine this is a support site,
and this is the ability to sell software through this site,
and warn you about safety and updates.
It has so many goals.
I remember when the team said we want to run experiments,
and I brought the group in and some of the managers,
and I said, do you know what you're optimizing for?
It was very funny because they surprised me.
They said, hey, Ronnie, we read some of your papers.
We know there's this term called OEC.
We decided the time on site is our OEC.
And I said, wait a minute, some of your main goals is support site.
Is people spending more time on a support site,
a good thing or a bad thing?
And then half the room thought that more time is better,
and half the room thought that more time is worse.
So an OEC is bad if directionally you can't agree on it.
That's a great tip.
Along these same lines, I know you're a big fan of platforms
and building a platform to run experiments versus just one-off
experiments.
Can you just talk briefly about that to give people a sense
of where they probably should be going with their experimentation approach?
Yeah, so I think the motivation is to bring the marginal cost of experiments down to zero.

So the more you self-service, go to a website, set up your experiment,
define your targets, define the metrics that you want.
People don't appreciate that the number of metrics
starts to grow really fast if you're doing things right.
And being, you could define 10,000 metrics that you wanted to be in your scorecard.
Big numbers.
So it was so big and people said it was computationally inefficient.
We broke them into templates so that if you were launching a UI experiment,
you would get this set of 2,000.
If you're doing a revenue experiment, you would get this set of 2,000 if you're doing.
So the point was build a platform that can quickly allow you to set up and run an experiment
and then analyze it.
I think one of the things that I will say at Airbnb is the analysis was relatively weak.
And so lots of data scientists were hired to be able to compensate for the fact
that the platform didn't do enough.
And this happens in other organizations too, where there's this trade-off.
But if you're building a good platform, invest in it so that more and more automation will allow
people to look at the analysis without the need to involve a data scientist.
We published a paper.
Again, I'll give it in the notes with this, you know, sort of a nice matrix of six axes
and how you move from crawl to walk to run to fly and what you need to build on those six axes.
So if, you know, one of the things that I do sometimes when I consult is I go into the
Oregon and say, where do you think you are on these six axes?
And that should be the guidance for what are the things you need to do next.
This is going to be the most epic show notes episode we've had yet.
Maybe a last question.
We talked about how important trust is to running experiments and how,
even though people talk about speed, trust ends up being most important.
Still, I want to ask you about speed.
Is there anything you recommend for helping people run experiments faster and get results
more quickly that they can implement?
Yeah. So I'll say a couple of things.
One is if your platform is good, then when the experiment finishes,
you should have a scorecard soon after.
Maybe it takes a day, but it shouldn't be.
Then you have to wait a week for the data scientist.
To me, this is the number one way to speed up things.
Now, in terms of using the data efficiently, there are mechanisms out there under the title
of variance reduction that help you reduce the variance of metrics so that you need less users
so that you can get results faster.
Some examples that you might think about are capping metrics.
So if your revenue metric is very skewed, maybe you say, well, if somebody purchased
over $1,000, let's make that $1,000.

At Airbnb, one of the key metrics, for example, is Knights Booked.
Well, it turns out that some people book tens of nights.
They're like an agency or something, hundreds of nights.
You may say, okay, let's just cap this.
It's unlikely that people book more than 30 days in a given month.
So that variance reduction technique will allow you to get statistically significant results faster.
And a third technique is called Cupid, which is an article that we published.
Again, I can give it in the notes, which uses the pre-experiment data to adjust the result.
And we can show that you get the result as unbiased, but with lower variance in health.
Hence, it requires fewer users.
Ron, is there anything else you want to share before we get to our very exciting lightning round?
No, I think we've asked a lot of good questions.
Hope people enjoy this.
I know they will.
Lightning round.
Lightning round.
Here we go.
I'm just going to roll right into it.
What are two or three books that you recommended most to other people?
There's a fun book called Calling Bullshit,
which is despite the name, which is a little extreme, I think.
For the title, it actually has a lot of amazing insights that I love.
And it embodies, in my opinion, a lot of the Twyman's Law of showing that things that are too extreme, your bullshit meter should go up and say, hey, I don't believe that.
So that's my number one recommendation.
There's a slightly older book that I love called Hard Facts,
Dangerous Half-Truths and Total Nonsense,
by the Stanford professors from the Graduate School of Business.
Very interesting to see many of the things that we grew up with as well understood,
turn out to have no justification.
And then a sort of stranger book, which I love sort of on the verge of psychology,
is called Mistakes Were Made, But Not By Me,
about all the fallacies that we fall into and the humbling results from that.
The titles of these are hilarious, and there's a common theme across all these books.
Next question, what is a favorite recent movie or TV show?
So I recently saw a short series called Chernobyl on The Disaster.
I thought it was amazingly well done.
Highly recommended.
Based on true events, as usual, there's some freedom for the artistic
movie that was kind of interesting.
At the end, they say this woman in the movie wasn't really a woman.
It was a bunch of 30 data scientists, about data scientists, 30 scientists,
that in real life presented all the data to the leadership of what to do.

I remember that.
Fun fact, I was born in Odessa, Ukraine, which was not so far from Chernobyl.
And I remember when my dad told me he had to go to work,
they called him into work that day to clean some stuff off the trees.
I think ash from the explosion or something.
It was like far away where I don't think we were exposed.
But yeah, we were in the vicinity.
That's pretty scary.
My wife thinks every time something's wrong with me,
she's like, oh, that must be a Chernobyl thing.
Okay, next question.
Favorite interview question you like to ask people when you're interviewing them?
So it depends on the interview, but I'll give you...
When I do a technical interview, which I do less of,
but one question that I love that is amazing how many people it throws away
for languages like C++ is, tell me what the static qualifier does.
And for multiple, you can do it for a variable, you can do it for a function.
And it is just amazing that I would say more than 50% of people that interview for an engineering job cannot get this and get it awfully wrong.
Definitely the most technical answer to this question yet.
Okay, what's a favorite recent product you've discovered that you love?
Blink cameras.
So a Blink camera is this small camera.
You stick in two AA batteries and it lasts for about six months.
They claim up to two years.
My experience is usually about six months, but it was just amazing to me how
you could throw these things around in the yard and see things that you would never know otherwise.
Some animals that go by, we had a skunk that we couldn't figure out how it was entering.
So I threw five cameras out and I saw where he came in.
Where did he come in?
He came in under a hole in the fence that was about this high.
I cannot, I have a video of this thing just squishing underneath.
We never would have assumed that it came from there, from the neighbor.
But yeah, these things have just changed.
And when you're away on a trip, it's always nice to be able to say,
I can see my house, everything's okay.
One point, we had a false alarm and the cops came in and had this amazing video of how they're entering the house and pulling the guns down.
You got to share that on TikTok.
That's good content.
Wow, okay.
But cameras, we'll set those up in my house ASAP.

Yes.
What is something relatively minor you've changed in the way your team's developed product that has had a big impact on their ability to execute?
I think this is something that I learned at Amazon, which is a structured narrative.
Amazon has some variants of this, which sometimes they go by the name of a six page or something.
But when I was at Amazon, I still remember that email from Jeff, which is,
no more PowerPoint, I'm going to force you to write a narrative.
I took that to heart and many of the features that the team presented instead of a PowerPoint,
you start off with a structured document that tells you what you need,
the questions you need to answer for your idea,
and then we review them as a team.
And Amazon, these were like paper-based.
Now it's all based on Word or Google Docs where people comment.
And I think the impact of that was amazing.
I think the ability to give people honest feedback and have them appreciate and have it stay after the meeting was in these notes on the document, just amazing.
Final question.
Have you ever run an A-B test on your life,
either your dating life, your family, your kids?
And if so, what did you try?
So there aren't enough units.
Remember, I said unique 10,000 of something to run through A-B tests.
I do, I will say a couple of things.
One is I try to emphasize to my family and friends and everybody this idea called the hierarchy of evidence.
When you read something, there's a hierarchy of trust levels.
If something is anecdotal, don't trust it.
If there was an experiment, it was observational, give it some bit of trust.
As you get more up and up to a natural experiment and a controlled experiments and multiple controlled experiments, your trust level should go up.
So I think that that's a very important thing that a lot of people miss when you see something in the news is where does it come from?
I have a talk that I've shared of all these observational studies that people made that were published and then somehow a controlled experiment was run later on and proved that it was directionally incorrect.
So I think there's a lot to learn about this idea of the hierarchy of evidence and share it with our family and kids and friends.
And there's a now, I think there's a book that's based on this is like how to read a book.
Well, Ronnie, the experiment of us recording a podcast, I think is 100% positive p-value, 0.0. Thank you so much for being here.
Thank you so much for inviting me and for great questions.
Amazing, I appreciate that.
Two final questions.

Where can folks find you online if they want to reach out?
And is there anything that listeners can do for you?
Finding me online is easy.
It's LinkedIn and what can people do for me?
Understand the idea of controlled experiments as a mechanism to make
the right data-driven decisions, use science.
Learn more by reading my book if you want.
Again, all proceeds go to charity.
And if you want to learn more, there's a class that I teach every quarter on Maven.
We'll put in the notes, how to find it, and some discount for people who
manage to stay all the way to the end of this podcast.
Yeah, that's awesome.
We'll include that at the top so people don't miss it.
So there's going to be a code to get a discount on your course.
Ronnie, thank you again so much for being here.
This was amazing.
Thank you so much.
Bye, everyone.
Thank you so much for listening.
If you found this valuable, you can subscribe to the show on Apple Podcasts, Spotify,
or your favorite podcast app.
Also, please consider giving us a rating or leaving a review,
as that really helps other listeners find the podcast.
You can find all past episodes or learn more about the show at LenniesPodcast.com.
See you in the next episode.