How to Build a Growth Engine with Product Experimentation

Lessons from Amazon, Microsoft, Facebook, Uber, Spotify, and Airbnb

Justin Warren, PivotNine

Anu Sharma, Statsig

Sponsored by Statsig

December 2022

v1.1

PivotNine

Contents

Preface

About the Authors

Justin Warren

Justin Warren is the Founder and Chief Analyst at PivotNine. He has worked with many well-known companies around the world, including ANZ Bank, IBM, NetApp, Nutanix, Pure Storage, Red Hat, Telstra, and VMware as well as a variety of startups including Atomist, Cloudistics, Elastifile, Hasura, Illumio, Isovalent, Manifold, Mattermost, and Solo.io, among others.

Anu Sharma

Anu Sharma is the Head of Product at Statsig, and a former Partner at Madrona Venture Group. Prior to joining Madrona, Anu led product development in EC2 for Amazon Web Services. She focuses on taking products from zero to one and is passionate about incorporating the scientific approach in product development.

About Statsig

Statsig is the leading experimentation and feature management platform helping businesses use data to move faster and build better products. Companies like Notion, Flipkart, Eventbrite, Ancestry, and Univision use Statsig to release features, automate experiments and get a comprehensive view of performance. Founded in 2021 by former Facebook engineers, Statsig supports thousands of experiments across billions of end users every day.

About PivotNine

PivotNine Pty Ltd is an analysis and consulting firm specialising in enterprise and developer technology.

We assist clients with the evaluation and selection of enterprise and developer technologies for datacenters, cloud, and information security. We conduct custom research on behalf of our clients to better inform their decision-making, and to ensure they are making an informed choice both now, and into the future.

PivotNine also assists vendors with product marketing, marketing strategy, and communicating their product benefits to enterprise buyers.

To learn more about becoming a PivotNine client, or to request reprint rights, contact sales@pivotnine.com.

Executive Summary

> *"You cannot invent without experimenting. If you want to invent more, you have to run more experiments per week, per month, per year, per decade."* — **Jeff Bezos**, Founder and Chairman, Amazon

Product experimentation is a proven method for quickly and efficiently scaling software products. While most are now familiar with the concept of A/B testing, leading organizations like Amazon, Facebook and Microsoft have taken the practice of product experiments to an entirely different level.

Rather than testing simple, largely cosmetic changes, they test more fundamental product concepts. Further, their experiments are performed at scale, simultaneously coordinating dozens or hundreds of experiments across teams.

We studied the evolution of product development at Amazon, Microsoft, Facebook, Uber, Spotify, and Airbnb, to uncover how they were able to build 'growth engines' based on product experimentation.

This report details the key features and benefits of a product experimentation approach. We cover how to evaluate product experimentation tools, how to develop a culture of experimentation, and how to avoid common mistakes along the way. This paper will provide a roadmap for building a growth engine in your own organization, based on what worked well—and what didn't—at our research companies.

Introduction

What is Product Experimentation?

Product experimentation is a system and process for conducting research about product performance in order to gather objective evidence of product success or failure. Without concrete evidence, it is difficult to decide which products and product features to invest in, and which to abandon.

Product experimentation helps us quickly identify new, beneficial features at low cost. Now every new idea is a success. Based on his experience at Microsoft, Ronny Kohavi found that "one-third prove effective, one-third have neutral results, and one-third have negative results."[1] Without product experimentation, it is difficult—if not impossible—to tell the difference.

This lack of evidence is costly, both in money but also in time. We learned in our research that teams can waste weeks, or even months, debating options if they lack the evidence required to make a decision.

It doesn't have to be this way.

Building a Growth Engine

Leading companies have, for many years now, adopted iterative development techniques using product experimentation tools and techniques with great success[2]. They have learned that it is better to have an inexact answer to the right question than an exact answer to the wrong question[3].

While we're all familiar with the concept of A/B testing*, the organizations we researched have taken the practice of product experiments to an entirely different level[See, e.g.: 4,5]. Rather than testing only fairly simple, largely cosmetic changes, they are able to test more fundamental product concepts. Their experiments are performed at scale, simultaneously coordinating dozens or hundreds of product teams' experiments.

They were able to focus investment where it matters most by investing in a culture of testing and evidence-based decision-making. Product choices that don't work are more obvious and can be quickly abandoned or changed[6].

Choices that do work are also more obvious, and investment can be allocated to these options. For people who get used to working this way, it can be a shock when they move to another organization that lacks the same culture[7].

Yet in every case, this culture didn't spring up fully formed. It was built. In some cases, it took multiple attempts before a product experimentation approach took hold. In others, what worked at first eventually needed to be replaced with new methods more suited to the now much larger company's needs.

There are lessons to be learned from every case, each one highlighting an important point we should keep in mind when building our own growth engine.

*A simple randomized controlled experiment where two close alternatives, such as different email subject lines or button colors, are shown to different groups of people, and the better performing option is ultimately chosen.

Case Studies

We studied the evolution of product development at six major companies: Amazon, Microsoft, Facebook, Uber, Spotify, and Airbnb. Each company has been able to build a 'growth engine' based on product experimentation.

We found that there have been two distinct eras in the development of product experimentation at these companies.

In the first era, starting around 1997 and ending around 2012, product experimentation was a relatively new idea and companies were inventing the field. The limiting factors were infrastructural. There were few people with product experimentation experience and no tools to support even small-scale experimenting. A lot of time was spent building the infrastructure to run experiments rather than running them.

In the second era, from 2012 onwards, expertise is more widespread, and commercial experimentation tools have begun to emerge. The limiting factor has shifted from infrastructure to culture. Any company can start doing product experiments relatively easily. All they require is the desire and commitment to do so.

Amazon

In 1997, Eric Benson, a software engineer at fledgling online retailer Amazon.com, tried out a new feature for the website: "Customers who bought X also bought Y". Unsurprisingly, it was a huge success. This quick, cheap experiment hinted at how gathering hard evidence of how a product performed, and doing it frequently, could lead to more business success.

The team at Amazon had been shipping new features constantly since launching the company in July 1995, yet it had little insight into how features were performing individually.

If Amazon wanted to evaluate the performance of a feature, a variant had to be shown to everyone accessing the site at once, changed, and then tested on everyone again. There was no way to show different variants of a feature to different audiences at the same time, or to separate out the influence of different features from one another.

And so, in 1998, Benson developed a proof-of-concept system to run online product experiments that could provide this level of separation. It was scrappy and rough, but it showed what could be done.

Benson proved it was possible to create a system that would automatically build experimental variants, deploy the variants to production, route traffic to appropriate variants, track usage through to the end of the session, extract logs and run analyses, and pull together the results.

It was a revolutionary idea for software development at the time[3].

While Benson left Amazon shortly afterwards, Amazon embraced this rigorous product experimentation approach wholeheartedly.

In 2001, JJ Jacobi, then Director of Website Analytics, led the effort to set up a centralized, 'full-service' *Weblab* team to do product experimentation for product teams. As a full-service team, Weblab provided all the experiment design, tooling, analysis, and decision-making services product teams might need, doing the work on their behalf.

> *"You cannot invent without experimenting. If you want to invent more, you have to run more experiments per week, per month, per year, per decade."* — **Jeff Bezos**, Founder and Chairman, Amazon

The practice of gathering evidence to inform product choices became a core part of how Amazon does product development that persists to this day. However, the centralized full-service model created a bottleneck.

The key to maximizing the rate of experimentation was to ensure that the cost of experiments was low. A centralized team, in high demand across the company, was driving costs up, not down. By 2003, as the company grew to over 5,000 employees, the full-service model became untenable.

Rather than funneling all experiments through a central team, Amazon moved to a *self-service* model. The company developed tools that allowed product teams to run their own experiments, independently, but without interfering with each other.

As Amazon's web infrastructure developed to enable more self-service, the cost of individual experiments continued to decline, even as more experiments were being run.

Moving to self-service created new challenges. Without the gatekeeping of a central team designing and running all the experiments, human biases within different teams could compromise the quality of experiments. Common biases such as overestimating the importance of small clusters, confirmation bias, and difficulty interpreting conditional probabilities can compromise the design of experiments or interpreting the results.

To counter these biases, Amazon instituted the Experiment Bar Raiser program in 2012. The program mandated pre-experiment checks* and post-experiment checks†. While the Bar Raiser process implemented important checks and balances, it also increased the cost of experiments.

As Amazon continues to grow, the complexity of operating a portfolio of products creates further challenges. Product decisions now need to account for new categories that encourage consumption compared to previous categories that simply encouraged purchases.

Weblab has now evolved to measure impact to *composite contribution profit* (an estimate of future cash flow) as distinct from simple order purchase sales as the company's topline metric.

Reflecting on Amazon's history, Bezos cited two key factors that forced the company's hand on inventing and experimentation: being in a fast-changing market that forced rapid exploration and invention, and Amazon's original cultural DNA.

As Bezos explained at a conference in 2005[8]:

> *"In our industry and given the DNA we have inside the company, we have attracted people who like to pioneer. We have the wrong culture to be close-followers, though it can be effective in some environments. Being customer-focused (vs. competitor-focused) is especially handy in our space because the rate of change in the online world is so rapid that "close following" doesn't work as well as it might in a more stable industry."*

Over time, experimentation has enabled Amazon to often 'just know the answer' to questions about what their customers want. There is no need to guess, nor is time wasted on evidence-free debate.

Microsoft

> *"The one great thing about controlled experiments is that once you experience the culture and see the value, you can't go back! Many young companies experiment but lose that ability as they grow larger."* — **Ronny Kohavi**, former Technical Fellow and Corporate Vice President, Analysis & Experimentation at Microsoft, 2005-2019.

*Such as a clearly stated objective and hypothesis, launch criteria, and running duration.
†Including a balanced executive summary, best reasons to launch and not launch, unexpected outcomes, and learnings.

In 2005, Ronny Kohavi joined Microsoft from Amazon and made a troubling discovery. Microsoft was still using an "old-school spec" of dev, test, and release running on three-year cycles with its products, including Microsoft Office. All product testing was offline and expensive, and the ratio of software developers to testers was 1:1. He saw years of development sunk into untested products as "a colossal waste of time and effort".

"My experience with Amazon Weblab was that most ideas fail when evaluated objectively in a controlled experiment," said Kohavi. "I saw an opportunity to shift teams to more agile development methodologies and evaluate ideas."

Kohavi proposed to build an online experimentation platform, called ExP. Microsoft's leadership saw it as an opportunity to reshape their culture.

"In March 2006 I was given the go-ahead to hire a small incubation team. In June 2006 coding started, and we ran our first two experiments a year later, in June 2007," Kohavi said.

"The goal for the ExP team was to accelerate the cultural change towards more use of data-driven decisions using experiments, not just to provide the technology or the platform. Our mission was to accelerate software innovation," Kohavi added.

In the beginning, anything that the ExP platform couldn't do, the ExP team did manually. Engineers identified repetitive jobs that data scientists performed and automated these tasks. The ExP team operated as a third-party startup, charging internal Microsoft teams to analyze their experiments. Their first experiments went live in June 2007 on the MSN homepage and on Windows Marketplace.

Yet changing culture is not a quick or easy undertaking, as Kohavi discovered. ExP's journey turned out to be painful and fraught with organizational risk.

"Because we were a small team, we had both functional challenges in being able to create a platform that scaled, and cultural challenges in convincing Microsoft to adopt experimentation," Kohavi said.

Bing had already built experimentation infrastructure for search when Ronny started the ExP team for other groups. Killed once in July 2006, ExP was born again in September 2006 with executive sponsorship from David Treadwell, who reported to Ray Ozzie and provided 'neutral ground' for ExP. However, ExP was merged back with Bing in 2009.

But Kohavi's persistence paid off.

"In 2014, when the group was split from Bing into an infrastructure team, it was clear that the value was understood. Our mission was to expand the usage to other groups, and specifically Office," he said.

"By the time I left in 2019, ExP was in use by every major product group at Microsoft."

Facebook

In 2009, Jay Parikh joined Facebook as Head of Engineering to help scale its engineering team and keep Facebook moving quickly.

"Outside of the integrity and security of user data, the number one thing that worried me from the beginning was slowing down," Parikh said. "Not slowing down was part investment in technology, and part investment in culture including hiring, incentives, and behaviors."

This was at a time when cloud-based Software-as-a-Service offerings were rare, and Facebook had become accustomed to writing its own tools. An absence of commercially available tools and the company's inclination to control its own destiny led Parikh to invest in building the product experimentation tools Facebook needed. Parikh believed the engineering tool chain was a vital component of keeping things moving.

"The experimentation tooling has a really important impact on helping an organization go faster as it gets bigger and the product complexity increases," Parikh said.

"You can discover the facts quickly when you have control over the data systems and the internal tools that you use to find, discover, and communicate the facts."

At the time, A/B testing was the primary method for making data-driven product decisions, but it was manual and slow. Teams had to pick metrics, then list members in test and control groups, then look up tables and summarize metrics, then determine differences between groups, and finally visualize the data, share the results, and decide which path to take.

A Facebook engineer explained how the chat application team lost time without data in 2011[2]:

> *"At the end of the day, the month we spent debating the product internally was a waste of time. We understood both sides of the debate within a day but had no data to resolve it."*

To meet this challenge, Facebook created several internal tools, including Gatekeeper, an in-house feature flagging platform, Quick Experiments, a tool for quickly trying out projects on a small subset of the Facebook user base, and Deltoid, an experiment analysis and reporting tool.

Quick Experiments helped teams to work autonomously and without getting in each other's way. Gatekeeper was used to generate quick data on how each feature performed before it shipped to everyone. Deltoid made the information accessible to everyone in a common format, and became a primary evidence source for feature rollout proposals. The faster product teams got data, the faster they shipped[2].

These tools transformed experimentation analysis from artisanal, hand-crafted projects into fully managed production systems, enabling experimentation volumes not seen in the industry before. Developing this strategic infrastructure and tooling became a matter of pride and attracted the best engineers in the company.

> *"There are tens of thousands of versions of Facebook running across the globe. The infrastructure it takes to pull that off is really, really cool. Every engineer can be running an experiment."* — **Jay Parikh**, Head of Engineering, Facebook, 2009-2020.

Like Amazon, Facebook found that providing tools that every product team could use independently was vital. Automation kept experimentation costs low even as more and more teams embraced the experimental approach to product development.

Keeping costs low meant more experiments could be run, and more product features were developed. Because the features were tested with rigorous experiments, product teams had the data to prove which features were working, and which ones had not worked and should be abandoned.

By sheer volume of experiments, Facebook executed Bezos' call for high volume, low cost experimentation better than Amazon itself.

Uber, Spotify, and Airbnb

> *"When running an experiment, the safe assumption is that unless extraordinary precautions are taken, it will run incorrectly."* — **George E.P. Box**[9]

Uber, Spotify and Airbnb were all formed after the major work of the first era of product experimentation had been completed. Product experimentation was well understood by engineers who had learned from the success of companies who built the infrastructure during the first era.

The world of tech had also changed substantially compared to the earlier era. Mobile devices, apps, and cloud-computing were all commonplace and expected ways of working. SaaS applications were abundant.

These second-era companies could start further down the path, skipping some of the growing pains experienced by the first-era pioneers. However, they encountered their own challenges.

Uber

Founded in 2009, Uber built its first experimentation platform, Morpheus, to implement both feature flagging and A/B testing in 2013. While it was initially successful, by 2020 Morpheus was becoming unwieldy and difficult to use[10].

> *"In early 2020, we took a deeper look at this (Morpheus) ecosystem. We discovered that a large percentage of the experiments had fatal problems and often needed to be rerun. Obtaining high-quality results required an expert-level understanding of experimentation and statistics, and an inordinate amount of toil (custom analysis, pipelining, etc.). This slowed down decision-making, and re-running poorly conducted experiments was common."*

More importantly, as feature flagging had become the norm and experimentation became a required dependency for mobile apps and backend services. Unreliable experimentation systems were causing outages, a major problem for a high-volume service with time-sensitive customers.

After rebuilding the platform in 2022, Uber transitioned over 2,000 developers, 15 integrated partner systems, 10 or more mobile apps, and more than 350 services to the new platform. 50,000 stale experiments running in Morpheus could finally be deprecated.

Spotify

Founded in 2006, Spotify initially used a small in-house team (called, simply, Analytics) and an ad-hoc, artisanal approach which "played around with various kinds of tests"[11].

In 2013, Spotify decided to build a more robust system for A/B testing, dubbed *ABBA*. The ABBA system proved useful, and "over time it was integrated into pretty much every aspect of Spotify"[11]. However, by 2017 the system had become a limiting factor in what Spotify wanted to achieve. ABBA only calculated a small number of metrics, and these metrics were not very sensitive, which meant "almost all" analysis was performed manually.

Spotify's requirements for the new experimentation platform were remarkably close to Uber's: making experiments easier to run and restart, reducing the toil involved in analyzing data, and providing a stable and reliable platform.

The new experimentation system, dubbed *The Experimentation Platform*, was rebuilt over the following two years. The new system added a host of functionality, including coordination of experiments (including domain exclusivity), a flexible experiment scheduler that catered for multiple teams working asynchronously, and validity checks[12].

Airbnb

Founded in 2008, Airbnb also iterated through two experimentation systems, just as Uber and Spotify had done.

Airbnb's first experimentation system, Trebuchet[13], was a fairly straightforward A/B testing tool written in Ruby introduced around 2010. It provided rudimentary testing capabilities, and lacked analysis or reporting features which limited its ability to scale.

Introduced in 2014, Airbnb's second system, the *Experimentation Reporting Framework* (ERF) aimed to avoid bias in assignment, make exposure logging reliable and automatic, and run production pipelines for automated analysis[14]. Starting with a few dozen experiments in 2014, ERF grew to run 500 concurrent experiments by 2017[15]. Airbnb continues to build on ERF, while also adding tools such as Superset for data exploration and reporting[16].

Building a Growth Engine

Our research has identified a 'fast-track' to building your own product experiment growth engine. We found that you can skip the activities of the first era almost completely, provided you heed the cautionary lessons of that time.

Starting here in the second era, you can adopt the same tools and techniques that the second era companies in our research adopted and tweak them to suit your own unique circumstances.

The following five step process can help us build a growth engine quickly:

1. Start experimenting at a small scale.
2. Build internal expertise and support.
3. Distribute knowledge in stages.
4. Expand systems for data sharing and coordination.
5. Tune the engine.

Start Experimenting

The first step in the process is to simply start doing experiments. In larger organizations, attempting to first build a broad consensus on how best to do product experiments is doomed to failure. Stable systems resist change* and experiments actively encourage controlled variability. In smaller organizations, there is no difference between getting started on a small scale and achieving a broad consensus.

Acquire existing commercial tools to run your experiments. As we saw in the case studies above, building your own in-house tools is a complex, time-consuming, and above all *risky* undertaking. There is no need to go through the pain of the first era companies yourself.

By starting small, you reduce the risk of making a poor choice of tool. If the initial tool choice doesn't work well, change it! This, in itself, is an example of running an experiment, gathering data, and making a decision based on that data.

At this early stage, changing tools doesn't affect many people or many experiments. You can start again, using the knowledge gained from the first set of experiments to make a better choice next time.

The important thing to do is to get started. Get a tool. Run experiments. Gather data and update your thinking. Learn by doing rather than wondering aloud.

Build Expertise and Support

The second stage builds on the first. In it, you run experiments, gather data, and build internal expertise in product experimentation.

Your choice of tool will guide your steps, as they have been built based on two decades of experience about what works and what doesn't. If you find yourself disagreeing with the tools, ask why that is. Are you trying to use them in ways they weren't designed to be used? Are you trying to paint a house with a hammer?

*This is what *stable* means.

Building expertise requires humility.

In this phase you are also building internal support for your approach. If the experiments are run well, you should be shipping more successful product features, and shipping them faster. Your team should be killing off failed ideas just as quickly, saving resources to invest in new ideas or doubling down on proven winners. There should be abundant proof that your approach is successful, and that kind of success invites imitation.

Expect to be approached by other teams who have noticed your success and want to learn your secrets. Now is not the time to be coy; now is the time to share widely.

Distribute Knowledge

The third stage is all about sharing your data, your tools, and your expertise. Look for and publicize evidence of your successes as well as your failures. Lead by example, providing a clear exemplar for others to imitate.

Share the data you've collected with the other teams that have come looking for it.[†] You want them to see what you are doing, and emulate it, using the same tools and methods. As more and more teams adopt, the desire to use product experiments will spread throughout your organization.

However, be alert to the danger of becoming a bottleneck. You don't want to emulate the experience of Amazon or Microsoft where a centralized team restricted the flow of experiments. Our research shows that *self-service* is the best approach to building a growth engine.

It may be tempting to position your team as elite experts with the unique ability to define how product experiments are conducted. Resist this temptation. Instead, strive to help people to help themselves.

Expand Data Sharing and Collaboration

At this stage, there should be widespread demand for product experiments across the company. To meet this demand, automation of standard tasks is vital.

Teams should be able to seek out data from previous experiments themselves. The data must therefore be readily discoverable as relevant given a question a team wants to answer. If relevant data doesn't already exist, teams should be able to use the experimenting tools to define, run, and analyze experiments themselves.

Some teams may lack the expertise to design high-quality experiments, or to reliably analyze the data they have collected. The tools should provide guidance towards known-good methods and help to educate the teams as they use the tools. For more complex requirements, a team of experts should be available as internal consults.

The goal here is to help people to help themselves, automating as much as possible, and training and teaching people to be self-sufficient where automation is unavailable. Success here is when many teams are running experiments autonomously, without getting in each other's way, and making use of data collected by others when it is available.

Tune the Engine

By this stage you will have a growth engine that is accelerating rapidly. Your goal now is to ensure that the engine doesn't tear itself apart.

[†]With appropriately designed access controls to preserve privacy and adherence to local laws, of course. The tool you select should readily support this.

To reduce waste, ensure that people can find answers to their questions using data that already exists, wherever possible. You should not be blindly running new experiments when you can simply look up the answers. Ensure that you are rewarding the behavior of using data to make decisions, not running experiments to gather data that already exists.

You will also want to ensure that you are running higher quality experiments. At this stage, you will already have a lot of data and expertise to draw from, and standard approaches to common problems will be developed. How can you ensure that everyone is using the most successful methods?

Friction and waste are what will slow you down at this stage. The goal is to tune the way your team is working to take best advantage of the data and tools you have, with as little effort as possible. How can you ensure that if one team already knows the answer, all of them do?

If someone develops a new, better way to run an experiment, how can we get everyone to quickly adopt this improved method?

At this final stage we are concerned less with technology and data than with people. All the evidence in the world won't help us if the culture and the mindset isn't there.

Learn More

We owe a great debt to those pioneers—Eric Benson, JJ Jacobi, Ronny Kohavi, Jay Parikh, and their teams—for forging a path through the wilderness. They suffered so that we don't have to. As with all great science, we can reach higher by standing on the shoulders of giants.

An active group of product experimentation experts and enthusiasts hang out in the Statsig Slack community at https://statsig.com/slack. They welcome new participants, and readers can enjoy personal access to some of the best minds in product experimentation. If you have questions, why not ask those at the forefront of building products with these techniques?

You can also explore the references used in this research. Most of these sources are publicly accessible, but if you have difficulty finding or accessing any of the references, you can contact the authors directly or email research@pivotnine.com and we will assist you.

References

1. Ron Kohavi & Stefan Thomke., (2017), "The Surprising Power Of Online Experiments," *Harvard Business Review*, https://hbr.org/2017/09/the-surprising-power-of-online-experiments.
2. (2021), "Building And Testing At Facebook," https://www.facebook.com/notes/10158791573022200/.
3. Jones, DM., (2020), *Evidence-Based Software Engineering: Based On The Publicly Available Data*, Knowledge Software, Ltd, http://www.knosof.co.uk/ESEUR/ESEUR.pdf.
4. Johari, R, Pekelis, L & Walsh, DJ., (2019), "Always Valid Inference: Bringing Sequential Analysis To A/B Testing," http://arxiv.org/abs/1512.04922.
5. Netflix Technology Blog., (2021), "It's All A/Bout Testing," https://netflixtechblog.com/its-all-a-bout-testing-the-netflix-experimentation-platform-4e1ca458c15.
6. Balasubramani, A., (2020), "Supporting Rapid Product Iteration With An Experimentation Analysis Platform," https://doordash.engineering/2020/09/09/experimentation-analysis-platform-mvp/.
7. Nichols, WR., (2019), "The End To The Myth Of Individual Programmer Productivity," *IEEE Software*, vol. 36, no. 5, pp. 71–75.
8. (2005), *Taking On The Challenge: Jeffrey Bezos, Amazon*, https://www.youtube.com/watch?v=WhnDvvNS8zQ.
9. Box, GEP, Hunter, JS & Hunter, WG., (2005), *Statistics For Experimenters: Design, Innovation, And Discovery* 2nd ed., Wiley-Interscience, Hoboken, N.J.
10. Sergey Gitlin, Krishna Puttaswamy, Luke Duncan, Deepak Bobbarjung & Arun Babu A S P., (2022), "Supercharging A/B Testing At Uber," https://www.uber.com/blog/supercharging-a-b-testing-at-uber/.
11. Johan Rydberg., (2020), "Spotify's New Experimentation Platform (Part 1)," https://engineering.atspotify.com/2020/10/spotifys-new-experimentation-platform-part-1/.
12. Johan Rydberg., (2020), "Spotify's New Experimentation Platform (Part 2)," https://engineering.atspotify.com/2020/11/spotifys-new-experimentation-platform-part-2/.
13. *Trebuchet*, Airbnb, https://github.com/airbnb/trebuchet.
14. Jan Overgoor., (2016), "Experiments At Airbnb," https://medium.com/airbnb-engineering/experiments-at-airbnb-e2db3abf39e7.
15. Parks, J., (2017), "Scaling Airbnb's Experimentation Platform," https://medium.com/airbnb-engineering/https-medium-com-jonathan-parks-scaling-erf-23fd17c91166.
16. AirbnbEng., (2017), "Superset: Airbnb's Data Exploration Platform," https://medium.com/airbnb-engineering/caravel-airbnb-s-data-exploration-platform-15a72aa610e5.

PivotNine