Top Challenges from the first Practical Online Controlled Experiments Summit

Somit Gupta (Microsoft)[1], Ronny Kohavi (Microsoft)[2], Diane Tang (Google)[3], Ya Xu (LinkedIn)[4], Reid Andersen (Airbnb), Eytan Bakshy (Facebook), Niall Cardin (Google), Sumitha Chandran (Lyft), Nanyu Chen (LinkedIn), Dominic Coey (Facebook), Mike Curtis (Google), Alex Deng (Microsoft), Weitao Duan (LinkedIn), Peter Forbes (Netflix), Brian Frasca (Microsoft), Tommy Guy (Microsoft), Guido W. Imbens (Stanford), Guillaume Saint Jacques (LinkedIn), Pranav Kantawala (Google), Ilya Katsev (Yandex), Moshe Katzwer (Uber), Mikael Konutgan (Facebook), Elena Kunakova (Yandex), Minyong Lee (Airbnb), MJ Lee (Lyft), Joseph Liu (Twitter), James McQueen (Amazon), Amir Najmi (Google), Brent Smith (Amazon), Vivek Trehan (Uber), Lukas Vermeer (Booking.com), Toby Walker (Microsoft), Jeffrey Wong (Netflix), Igor Yashkov (Yandex)

ABSTRACT

Online controlled experiments (OCEs), also known as A/B tests, have become ubiquitous in evaluating the impact of changes made to software products and services. While the concept of online controlled experiments is simple, there are many practical challenges in running OCEs at scale. To understand the top practical challenges in running OCEs at scale and encourage further academic and industrial exploration, representatives with experience in large-scale experimentation from thirteen different organizations (Airbnb, Amazon, Booking.com, Facebook, Google, LinkedIn, Lyft, Microsoft, Netflix, Twitter, Uber, Yandex, and Stanford University) were invited to the first Practical Online Controlled Experiments Summit. All thirteen organizations sent representatives. Together these organizations have tested more than one hundred thousand experiment treatments last year. Thirty-four experts from these organizations participated in the summit in Sunnyvale, CA, USA on December 13-14, 2018.

While there are papers from individual organizations on some of the challenges and pitfalls in running OCEs at scale, this is the first paper to provide the top challenges faced across the industry for running OCEs at scale and some common solutions.

1. INTRODUCTION

The Internet provides developers of connected software, including web sites, applications, and devices, an unprecedented opportunity to accelerate innovation by evaluating ideas quickly and accurately using OCEs. At companies that run OCEs at scale, the tests have very low marginal cost and can run with thousands to millions of users. As a result, OCEs are quite ubiquitous in the technology industry. From front-end user-interface changes to backend algorithms, from search engines (e.g., Google, Bing, Yandex) to retailers (e.g., Amazon, eBay, Etsy) to media service providers (e.g. Netflix, Amazon) to social networking services (e.g., Facebook, LinkedIn, Twitter) to travel services (e.g., Lyft, Uber, Airbnb, Booking.com), OCEs now help make data-driven decisions [7, 10, 61, 76, 12, 27, 30, 40, 41, 44, 51, 58].

1.1 First Practical Online Controlled Experiments Summit, 2018

To understand the top practical challenges in running OCEs at scale, representatives with experience in large-scale experimentation from thirteen different organizations (Airbnb, Amazon, Booking.com, Facebook, Google, LinkedIn, Lyft, Microsoft, Netflix, Twitter, Uber, Yandex, and Stanford University) were invited to the first Practical Online Controlled Experiments Summit. All thirteen organizations sent representatives. Together these organizations have tested more than one hundred thousand experiment treatments last year. Thirty-four experts from these organizations participated in the summit in Sunnyvale, CA, USA on December 13-14, 2018. The summit was chaired by Ronny Kohavi (Microsoft), Diane Tang (Google), and Ya Xu (LinkedIn). During the summit, each company presented an overview of experimentation operations and the top three challenges they faced. Before the summit, participants completed a survey of topics they would like to discuss. Based on the popular topics, there were nine breakout sessions detailing these issues. Breakout sessions occurred over two days. Each participant could participate in at least two breakout sessions. Each breakout group presented a summary of their session to all summit participants and further discussed topics with them. This paper highlights top challenges in the field of OCEs and common solutions based on discussions leading up to the summit, during the summit, and afterwards.

1.2 Online Controlled Experiments

Online Controlled Experiments, A/B tests or simply experiments, are widely used by data-driven companies to evaluate the impact of software changes (e.g. new features). In the simplest OCE, users are randomly assigned to one of the two variants: Control (A) or Treatment (B). Usually, Control is the existing system and Treatment is the system with the new feature, say, feature X. User interactions with the system are recorded and from that, metrics computed. If the experiment was designed and executed correctly, the only thing consistently different between the two variants is feature X. External factors such as seasonality, impact of other feature launches, or moves by the competition, are evenly distributed between Control and Treatment, which means that we

[1] Organizers email: somit.gupta@microsoft.com

[2] ronnyk@microsoft.com

[3] diane@google.com

[4] yaxu@linkedin.com

can hypothesize that any difference in metrics between the two groups can be attributed to either feature X or due to chance resulting from the random assignment to the variants. The latter hypothesis is ruled out (probabilistically) using statistical tests such as a t-test [21]. This establishes a causal relationship (with high probability) between the feature change and changes in user behavior, which is a key reason for the widespread use of controlled experiments.

1.3 Contribution

OCEs rely on the same theory as randomized controlled trials (RCTs). The theory of a controlled experiment dates back to Sir Ronald A. Fisher's experiments at the Rothamsted Agricultural Experimental Station in England in the 1920s. While the theory is simple, deployment and evaluation of OCEs at scale (100s of concurrently running experiments) across variety of web sites, mobile apps, and desktop applications presents many pitfalls and research challenges. Over the years some of these challenges and pitfalls have been described by authors from different companies along with novel methods to address some of those challenges [16, 23, 43, 48, 50, 61]. This is the first time that OCE experts from thirteen organizations with some of the largest scale experimentation platforms have come together to identify the top challenges facing the industry and the common methods for addressing these challenges. This is the novel contribution of this paper. We hope that this paper provides researchers a clear set of problems currently facing the industry and spurs further research in these areas in academia and industry. It is our wish to continue this cross-company collaboration to help advance the science of OCEs.

Section 2 presents an overview of top challenges currently faced by companies across the industry. Later sections discuss specific problems in more detail, how these problems have an impact on running OCEs in different products, and common solutions for dealing with them.

2. TOP CHALLENGES

Software applications and services run at a very large scale and serve tens to hundreds of millions of users. It is relatively low cost to update software to try out new ideas. Hundreds of changes are made in a software product during a short period of time. This provides OCEs a unique set of opportunities and challenges to help identify the good changes from the bad to improve products.

Our compiled list of top challenges comes from three sources: the pre-summit survey to collect the list of topics to discuss, the top challenges presented by each company at the start of the summit, and the post-summit survey on top takeaways and list of topics to discuss in future. The relative order of prevalence remained roughly the same across these sources. These top challenges reflect the high level of maturity and scale of OCE operations in these companies. There is literature out there on the challenges and pitfalls some of these companies faced and solved during the early years of their operations [32]. The challenges mentioned here are at the frontier of research in academia and industry. While there are some solutions to existing challenges, there is a lot of scope for further advancement in these areas.

1. **Analysis**: There are many interesting and challenging open questions for OCE results analysis. While most experiments in the industry run for 2 weeks or less, we are really interested in detecting the long-term effect of a change. How do long-term effects differ from short-term outcomes? How can we accurately measure those long-term factors without having to wait a long time in every case? What should be the *overall evaluation criterion* (OEC) for an experiment? How can we make sure that the OEC penalizes things like clickbaits that increase user dissatisfaction? While there are methods to test an OEC based on a set of labelled experiments [24], how to best collect such set of experiments for evaluation of the OEC and other metrics? While, there are models for estimating the long-term value (LTV) of a customer that may be a result of a complex machine learning model, can we leverage such models to create OEC metrics? Once we have OEC metrics and a Treatment improves or regresses the OEC metric, how can we best answer why the OEC metric improved or regressed and uncover the underlying causal mechanism or root cause for it?

   Running experiments at a large scale introduces another set of issues. It is common for large products serving millions of users to have 100s of experiments running concurrently, where each experiment can include millions of users. For products running so many experiments, most of the low-hanging fruit get picked quickly and many Treatments may then cause a very small change in OEC metrics. It is important to detect these types of changes. A very small change in a per-user metric may imply a change of millions of dollars in product revenue. How can we best increase the sensitivity of OECs and other experiments metrics without hurting the accuracy of these metrics to discern between good and bad Treatments [18, 42, 75]? If we are running 100s of experiments concurrently how do we handle the issue of interaction between two treatments? How can we learn more from analyzing multiple experiments together and sharing learnings across experiments? For a product with millions of users, there are many ways to segment users. Even a small fraction of users is very significant. Just understanding the average treatment effect on the entire population is not enough. How can we best identify heterogenous Treatment effects in different segments?

2. **Engineering and Culture:** Culture is the tacit social order of an organization: It shapes attitudes and behaviors in wide-ranging and durable ways. Cultural norms define what is encouraged, discouraged, accepted, or rejected within a group [35]. How do we build a culture to one that uses OCEs at scale to ensure we get a trustworthy estimate of the impact of every change made to a product and bases ship decisions on the outcome of OCEs [46]?

   Engineering systems and tools are critical aspects to enable OCEs at scale. What are some good development practices, data logging and data engineering patterns that aid trustworthy experimentation at scale?

3. **Deviations from Traditional A/B Tests:** Traditional A/B tests depend on a stable unit treatment value assumption(SUTVA) [39], that is, the response of any experiment unit (user) under treatment is independent of the response of another experiment unit under treatment. There are cases where this assumption does not hold true, such as network interactions or interactions between multiple experiments. If this issue is ignored, we may get a biased estimate of the treatment effect. How can we detect such deviation? Where deviations are unavoidable,

what is the best method to obtain a good estimate of the treatment effect?

4. **Data quality:** Trustworthiness of the results of an OCE depend on good data quality. What are some best practices to ensure good data quality? While the sample Ratio Mismatch test is a standard industry test to indicate data quality issues in OCEs [13, 22, 23, 45], what are other critical data quality tests to perform during OCE results analysis?

3. ESTIMATING THE LONG-TERM EFFECT

3.1 Problem

Though it only takes a few days to change a part of a software product or service, the impact of that change may take a long time to materialize in terms of key product indicators (KPIs) and can vary across products and scenarios. This makes it challenging to estimate the long-term impact of a change. For instance, the impact of a change of ranking results in an online travel service on customer satisfaction may not be fully understood until customers stay in a vacation rental or hotel room months after booking. Increasing the number of ads and hence decreasing their quality on a search results page may bring in more revenue in the first few weeks, but might have the opposite impact months later due to user attrition and users learning that ad results are less useful and ignoring them [38]. Investing in user retention and satisfaction through better user experience can be more beneficial over the long term than what short term measurements indicate. Introduction of clickbaits on a content provider service may cause increase in clicks due to the novelty effect but may induce larger dissatisfaction in the long term as users learn about poor content quality. Further, in two-sided markets [71], some changes like pricing in ads, ride-sharing services, or home-sharing services may introduce a market effect with a shift in either demand or supply in the eco-system and it may take a long time before the market finds a new equilibrium.

3.2 Common Solutions and Challenges

3.2.1 Long-term Experiments or Holdouts

Running experiments for a long duration is not usually a good answer. Most software companies have very short development cycles for planning, developing, testing, ultimately shipping new features. Short development cycles enable companies to be agile and quickly adapt to customer needs and the market. Long testing phases for understanding the impact of changes could harm a company's agility and are not usually desirable.

Another option is a long-term holdout group consisting of a random sample of users who do not get updates. This holdout group acts as the Control against the set of features shipping to everyone else. This option usually incurs a lot of engineering cost. The product development team must maintain a code fork that is not updated for a long time. All upstream and downstream components to this code must support this fork as well. This still does not solve the challenges of non-persistent user tracking and network interactions described below.

In many products and services, the first visit and subsequent visits of users is tracked using a non-persistent user identifier, like a random GUID [72] stored in a browser cookie. This way of tracking users is not very durable over a long time as users churn their cookies and we are left with tracking a biased sample of all users exposed to the variants [23]. Further, a user may access the same service from multiple devices, and the user's friends and family may access the same service. As time goes on, a user or their friends or family may be exposed to both the treatment and control experience during an experiment, which dilutes the impact of the treatment being measured in the experiment.

There is some value in running experiments a little longer when we suspect that there is a short-term novelty or user learning effect. At Microsoft, while most experiments do not run for more than two weeks, it is recommended to run an experiment longer if novelty effects are suspected and use data from the last week to estimate the long-term treatment effect [23]. At Twitter, a similar practice is followed. An experiment at Twitter may run for 4 weeks and data from last two weeks is analyzed. If a user exposed in the first two weeks does not appear in the last two weeks, values are imputed for that user when possible (like imputing 0 clicks). However, it may not be possible to impute values for metrics, like ratio or performance metrics.

3.2.2 Proxies

Good proxies that are predictive of the long-term outcome of interest are commonly used to estimate the long-term impact. For instance, Netflix has used logistic regression to find good predictors for user retention. Netflix also used survival analysis to take censoring of user data into account. LinkedIn created metrics based on a lifetime value model. For treatments that effect the overall market, Uber found some macro-economic models to be useful in finding good proxies. There can be downsides to this approach as correlation may not imply causation, and such proxies could be susceptible to misuse, where a treatment may cause an increase in the proxy metric, but ends up having no effect or regression in the long-term outcome. It may be better to develop a mental causal structure model to find good proxies. Bing and Google have found proxies for user satisfaction and retention by having a mental causal structure model that estimates the utility of an experience to users [36, 49].

3.2.3 Modeling User Learning

Another approach followed by Google is to explicitly model the user learning effects using some long duration experiments [38]. In long duration experiments, there are multiple and exclusive random samples of users exposed to the treatment. One group is exposed to the treatment from the start of the experiment. A second group has a lagged start, being exposed to the treatment at some point after the start, and so on. Comparing these groups a day after the second group is exposed to the treatment provides an estimate of user learning from the treatment. Google also used cookie-cookie day randomization to get an estimate of user learning for any duration (in days) since the experiment started. In these experiments and in the subsequent analysis, the authors carefully designed the experiments and did careful analysis to ensure that they were not seeing many confounding effects (e.g., other system changes, system learning, concept drift, as well as selection bias issues due to cookie churn/short cookie lifetimes). They took this information and modeled user learning as an exponential curve, which allowed them to predict the long-term outcome of a treatment using the short-term impact of the treatment directly measured in the experiment and the prediction of the impact of the treatment on user learning.

3.2.4 Surrogates

Surrogate modeling is another way to find good estimates of long-term outcome. A statistical surrogate lies on the causal path between the treatment and the long-term outcome. It satisfies the condition that treatment and outcome are independent conditional on the statistical surrogate. You can use observational data and experiment data to find good surrogates. Even if no individual proxy satisfies the statistical surrogacy criterion, a high-dimensional vector of proxies may collectively satisfy the surrogacy assumption [8]. Having a rich set of surrogates reduces the risk of affecting only a few surrogates and not the long-term outcome. Facebook used this approach with some success to find good surrogates of the 7-day outcome of an experiment by just using 2-3-day experiment results. They used quantile regression and a gradient-boosted regression tree to rank feature importance. Note that there is still a risk that having too many surrogates for the long term may make this approach less interpretable.

4. OEC: OVERALL EVALUATION CRITERION METRIC

4.1 Problem

One key benefit of evaluating new ideas through OCEs is that we can streamline the decision-making process and make it more objective. Without understanding the causal impact of an idea on customers, the decision-making process requires a lot of debate. The proponents and opponents of the idea advance their arguments regarding the change only relying on their own experience, recall, and interpretation of certain business reports and user comments. Eventually the team leader makes a call on whether to ship the idea. This style of decision making is based on the HiPPO (Highest Paid Person's Opinion) [37] and is fraught with many cognitive biases [2]. To help change HiPPO decisions to more objective, data-driven decisions based on the causal impact of an idea from customer response [5], we recommend establishing the OEC for all experiments on your product.

Not all metrics computed to analyze the results of an experiment are part of the OEC. To analyze experiment results, we require different types of metrics [22]. First, we need to know if the results of an experiment are trustworthy. A set of data quality metrics, like a sample ratio, help raise red flags on critical data quality issues. After checking the data quality metrics, we want to know the outcome of the experiment. Was the treatment successful and what was its impact? This set of metrics comprise the OEC. In addition to OEC metrics, we have found that there is a set of guardrail metrics which are not clearly indicative of success of the feature being tested, but metrics that we do not want to harm. The remaining bulk of the metrics for an experiment are diagnostic, feature or local metrics. These metrics help you understand the source of OEC movement (or the lack of).

It is hard to find a good OEC. Here are a few key properties [19, 24, 55] to consider. First, a good OEC must be indicative of the long-term gain in key product indicators (KPIs). At the very least make it directionally accurate in estimating the impact on the long-term outcome. Second, OEC must be hard to game and it should incentivize the right set of actions in the product team. It should not be easy to satisfy the OEC by doing the wrong thing. For instance, if the OEC is limited to a part or feature of the product, you may be able to satisfy the OEC by cannibalizing other parts or features. Third, OEC metrics must be *sensitive*. Most changes that impact the long-term outcome should also have a statistically significant movement in OEC metrics so it is practical to use the OEC to distinguish between good and bad changes to the product. Fourth, the cost of computing OEC metrics cannot be too expensive. OEC metrics must be computed for 100s of the experiments and be run on millions of users each and every week. Methods that involve costly computation or costly procedures like human surveys or human judges may not scale well. Fifth, OEC metrics must account for a diverse set of scenarios that may drive the key product goals. Finally, OEC should be able to accommodate new scenarios. For instance, direct answers to queries like current time would provide a good user experience in a search engine, but if you only base the OEC metrics on clicks, those metrics will miss this scenario.

4.2 Common Solutions and Challenges

4.2.1 Search vs. Discovery

Measuring the success of a search experience in search engines has been a research subject for a long time in academia and for many products including Bing, Google and Yandex. It is well established that metrics, like queries per user, cannot be a good OEC because queries per user may go up when search ranking degrades. Sessions per user or visits per user are considered better OEC metrics [49]. In general there is an appreciation of focusing on HEART (Happiness, Engagement, Adoption, Retention, and Task success) metrics for the OEC and use PULSE (Page views, Uptime, Latency, Seven-day active users [i.e., the number of unique users who used the product at least once in the last week], and Earnings) metrics as your guardrail metrics [62]. Over time, different methods have been proposed to measure HEART metrics in search engines and other goal-directed activities [36, 54].

It is still challenging to find metrics similar to HEART metrics to work for discovery- or browsing-related scenarios, like news articles shown on the Edge browser homepage, or Google mobile homepage, or Yandex homepage. The challenge is to understand user intent. Sometimes users will come with a goal-oriented intent and would like to quickly find what they are looking for. Other times users may have a more browsing or discovering-new-information intent where they are not looking for something specific but just exploring a topic. In this case it is not clear if lack of a click on an article link with a summary snippet can be viewed as a negative experience or positive because users got the gist of the article and did not have to click further. Further the two intents (goal-oriented and browsing) can compete. If a user came with a goal-oriented intent but got distracted and ended up browsing more, it may cause dissatisfaction in the long term.

4.2.2 Product Goals and Tradeoffs

OEC metrics usually indicate improvement in product KPIs or goals in the long term. This assumes that product goals are clear. This is not a trivial problem. It takes a lot of effort and energy to have clarity on product goals and strategy alignment across the entire team. This includes decisions like defining who the customer is and how best to serve them. Further, your team must also create a monetization strategy for the product. In absence of such clarity, each sub team in the product group may set their own goals that may not align with other teams' or corporate goals.

Even after the product goals are clear, in most companies you end up with a handful of key metrics of interest. It is challenging how to weigh these metrics relative to each other. For instance, a product may have goals around revenue and user happiness. If a feature increases user happiness but losses revenue, in what case is it

desirable to ship this feature? This decision is often made on a case-by-case basis by leadership. Such an approach is susceptible to a lot of cognitive biases, and also may result in an incoherent application of the overall strategy. Some product teams like at Bing, tried to come up with weights on different goals to make this tradeoff align with the overall product strategy and help ensure that consistent decision-making processes are used across multiple experiments.

4.2.3 Evaluating Methods

We mentioned that OEC metrics help decision making more objective. For it to be widely adopted, it is important to establish a process to evaluate any changes to the OEC metrics. In some cases, there may be an expert review team that examines changes to the OEC and ensure that it retains the properties of a good OEC. To make this process even more objective, we can have a corpus of past experiments widely seen to have a positive, negative or neutral impact. Changes to the OEC were evaluated on this corpus to ensure sensitivity and directional correctness [24]. Microsoft and Yandex successfully used this approach to update OEC metrics. The challenge here is to create a scalable corpus of experiments with trustworthy labels.

Other approaches include doing degradation experiments, where you intentionally degrade the product in a treatment and evaluate if the OEC metrics can detect this degradation. One well known example are experiments that slow down the user experience performed at Microsoft and Google [63, 64]. This is also a good thought exercise to go through while designing and evaluating OEC metrics to ensure that they are not gameable.

4.2.4 Using Machine Learning in Metrics

Some product teams tried to incorporate machine learning models (MLMs) to create a metric. For instance, using sequences of user actions to create a score metric based on the likelihood of user satisfaction [36, 54, 57] or creating more sensitive OEC metrics by combining different metrics [25, 42, 59]. Also, good proxies for long-term outcomes are often used to find good OEC metrics. This area of experimentation is relatively new. Many product teams are carefully trying to test these methods in limited areas. These methods are more commonly used in mature product areas, like search, where most of the low-hanging fruit is picked and we need more complex models to detect smaller changes. For new products, it is usually better to use simple metrics as the OEC.

There are some concerns with using machine learning models to create metrics. MLM based metrics can be harder to interpret and can appear as a blackbox, which reduces trustworthiness and makes it hard to understand why a metric may have moved. Refreshing MLMs by training them on most recent data may lead to an abrupt change in the metric that would hard to account for. If the MLM is being refreshed while an experiment is running, it can create bias in the metric. Further, there are concerns these metrics are easily gamed by optimizing for the underlying model that created these metrics, which may or may not lead to improvement in the long-term outcome of interest.

5. HETEROGENIETY IN TREATMENT EFFECTS (HTE)

5.1 Problem

Without loss of generality, we consider the case that there is only one treatment and one control. Under the potential outcome framework, $(Y(1), Y(0))$ is the potential outcome pairs and $\tau = Y(1) - Y(0)$ is the individual treatment effect.

The primary goal of an A/B test is to understand the average treatment effect (ATE), $E(\tau)$. Although it is obvious that knowing individual effect is ideal, it is also impossible as we cannot observe the counterfactual. The closest thing is the conditional average treatment effect (CATE) [74], $E(\tau|X)$, where X is some attribute or side information about each individual that is not affected by the treatment. This makes CATE the best regression prediction of individual treatment effect τ based on X.

Attributes X can be either discrete/categorical or continuous. Categorical X segments the whole population into subpopulations, or segments. In practice, the industry almost entirely uses categorical attributes. Even continuous attributes are made discrete and considered ordered categorical segments.

Perhaps the most interesting cases are when treatment moves the same metric in different directions, or when the same metric has statistically significant movement in one segment but not in another segment. Assume, for a given segment, say market, a metric moves positively for some markets but negatively for another, both highly statistically significant. Making the same ship decision for all segments would be sub-optimal. Such cases uncover key insights about the differences between segments. Further investigation is needed to understand why the treatment was not appreciated in some markets and identify opportunities for improvement. In some cases adaptive models can be used to fit different treatments on different types of users [6, 52, 53, 77].

However, most common cases of HTE only show difference in magnitude, not direction. Knowledge of these differences can be valuable for detecting outlier segments that may be indicative of bugs affecting a segment, or for encouraging further investment into different segments based on results.

5.2 Common Solutions and Challenges

5.2.1 Common Segments

It is a very common practice to define key segments based on product and user knowledge. Where possible, it is preferred to define segments so that the treatment does not interact with the segment definition to avoid bias.

Here are some of commonly defined segments for many software products and services:

1. **Market/country:** Market is commonly used by all companies with global presence who are running experiments and shipping features across different markets. When there are too many markets, it is useful to put them into larger categories or buckets like markets already with high penetration and growing markets or markets clustered by language.
2. **User activity level:** Classifying users based on their activity level into heavy, light and new users can show interesting HTE. It is important to have this classification based on data before the experiment started to avoid any bias.
3. **Device and platform:** Today most products have both desktop and mobile application. We can test most backend server-side features across devices and platforms. With device and platform fragmentation, it is getting harder to eliminate bugs for all devices and

platforms. Using device and platform segments in A/B testing is essential to flag potential bugs using live traffic. For example, in a recent experiment, a feature of the Outlook mobile app was moving key metrics on all Android devices except a few versions, which indicated further investigation was needed. Device and platforms also represent different demographics. Many studies show a difference between iOS users and Android users.

4. **Time and day of week:** Another common segment used is time. Plotting the effects delta or percent delta by day can show interesting patterns, such as the weekday and weekend effect, reveal a novelty effect [13], and help flag data quality issues.
5. **Product specific segments:** LinkedIn segmented users by normal user and recruiter. On Twitter, some handles can belong to a single user, so it is useful to segment Twitter handles by primary or secondary account. For Netflix, network speed and device types have proved to be good segments. Airbnb has found that segments of customers based on whether they have booked before and based on from where they first arrived on Airbnb site are useful.

5.2.2 Methodology and Computation

Our community recognizes a lot of recent work from both academia and industry. The most common mental model is the linear model with a first-order interaction term between treatment assignment and covariates X: $Y = \theta + \delta T + \beta \times T \times X + \epsilon$.

Most useful segments used by the community are categorical, so the linear model suffices. There is consensus that the first-order treatment effect adjustment by a single covariate, such as a segment of one categorical variable, is the most actionable. One active area of research is adapting more MLMs for identifying HTE [74].

Nevertheless, there are a lot of outstanding challenges:

1. **Computation scale:** Because A/B tests routinely analyze hundreds or thousands of metrics on millions of experiment units (users), the resources and time spent on an automatically scheduled analysis cannot be too much to ensure that results are not delayed and are not too expensive to generate. There is a desire to use a simple algorithm directly formulated using sufficient statistics, instead of using individual-unit level data.
2. **Low Signal Noise Ratio (SNR):** A/B testing is already dealing with low power to estimate the average treatment effect. Learning HTE is even harder than learning ATE because of the reduced sample sizes in each subpopulation.
3. **Multiple Testing Problem** [66]**:** There is a severe multiple testing problem when looking at many metrics, and many possible ways to segment the population. This issue, along with low SNR further complicates HTE estimations.
4. **Interpretable and memorable results:** Most experimenters are not experts in statistics or machine learning. You must have concise and memorable result summaries to facilitate experimenters to act.
5. **Absolute vs. Relative:** While determining the HTE, you must decide whether you will use absolute CATE or relative CATE (as a percentage of average value of the metric in control). In many cases it makes sense to use the relative CATE as the baseline or the average value of a control metric can be very different for different segments, like different countries. Use a relative CATE to normalize the treatment effect in different segments.

To tackle these challenges, there are common approaches companies take.

1. Separate on-demand and scheduled analysis. For on-demand analysis, people are willing to spend more resources and wait longer to get results. For this kind of one-off analysis, linear regression with sparsity (L1 and elastic net) and tree-based algorithms, like causal tree, are very popular. Double ML also gained a lot of attention recently [14].
2. Because of the challenge of low SNR and multiple testing, sparse modeling is a must. Even if the ground truth is not sparse, there are limited resources that experimenters can spend on learning and taking actions based on HTE. Sparse modeling forces concise results.
3. To make results memorable, when certain segment has many values, markets might have a lot of values, it is desired to merge those values based on a common effect. For instance, the effect might be different for Asian markets compared to rest of the world. Instead of reporting market HTE and list treatment effect estimates for individual markets, it is better to merge Asian markets and the rest of the world, and report only two different effect estimates. Algorithms that can perform regression and clustering is preferred in these cases, including Fused Lasso [69] and Total Variation Regularization.

5.2.3 Correlation is not Causation

Another difficulty in acting based on HTE results is more fundamental: HTE results are not causal, only correlational. HTE is a regression to predict individual treatment effect based on covariates X. There is no guarantee that predictor X explains the root cause of the HTE. In fact, when covariates X are correlated, there might be even issues like collinearity. For example, we may find HTE in devices showing iOS users and Android users have different effect. Do we know if device is the reason why the treatment effects are different? Of course not. iOS and Android users are different in many ways. To help experimenters investigate the difference, an HTE model that can adjust the contribution of devices by other factors would be more useful. Historical patterns and knowledge about whether investigating a segment X helped to understand HTE of a metric M could provide extra side information.

6. DEVELOPING EXPERIMENTATION CULTURE

6.1 Problem

Culture is the tacit social order of an organization. It shapes attitudes and behaviors in wide-ranging and durable ways. Cultural

norms define what is encouraged, discouraged, accepted, or rejected within a group [35]. There is a big challenge in creating an experiment-driven product development culture in an organization.

Cultural change involves transformation of an organization through multiple phases. There may be hubris at first, where every idea of the team is considered a winner. Then there may be introduction of some skepticism as the team begins experimentation and its intuition gets challenged. Finally, a culture develops where there is humility about our value judgement of different ideas, and better understanding of the product and customers [3].

It is well known that our intuition is a poor judge for the value of ideas. Case studies at Microsoft showed a third of all ideas tested through an OCE succeed in showing statistically significant improvements in key metrics of interest, and a third showed statistically significant regressions. Similar results have been noted by many major software companies [3, 17, 28, 47, 56, 60]. Yet it can be hard to subject your idea to an OCE and receive negative feedback, especially when you have spent a lot of time working on implementing it and selling it to your team. This phenomenon is not unique to the software industry. It is generally referred to as Semmelweis Reflex, based on the story of the long and hard transition of mindset among doctors about the importance of hygiene and having clean hands and scrubs before visiting a patient [65]. It takes a while to transition from a point where negative experiment results feel like someone telling you that your baby is ugly. You must enact a paradigm shift to put your customers and business in focus and listen to customer responses. At that point, negative experiment results are celebrated as saving customers and your business from harm. Note that not only bad ideas (including bloodletting [11]) appear as great ideas to a human mind, we are also likely to discount the value of great ideas (including good hand hygiene for doctors [65]). There are cases where an idea that languished in the product backlog for months as no one thought it was valuable turns out to be one of the best ideas for the product in its history [51].

A culture of working together towards the common goal of improving products through OCEs amplifies the benefits of controlled experimentation at scale [32]. This paves the way for frictionless integration of OCEs into the development process, and makes it easy to run an OCE to test an idea, get automated and trustworthy analysis of this experiment quickly, and interpret the results to take the next step: ship the feature, iterate, or discard the idea. A strong experimentation culture ensures that all changes to the product are tested using OCEs and teams benefit from OCEs discovering valuable improvements while not degrading product quality. It allows you to streamline product development discussions so everyone understands the OEC for the product and can take an objective decision to ship a feature based on the impact on the OEC metric. This gives developers freedom to build and test different ideas with minimum viable improvements without having to sell the entire team on the idea beforehand. And allows the team to make future decisions to invest in a product area based on changes to the OEC metric due to features seen in that area.

6.2 Common Solutions and Challenges

There are many cultural aspects to adoption of OCEs at scale to have a trustworthy estimate of the impact of every change made to a product.

6.2.1 Experimentation Platform and Tools

First, we need to make sure that the experimentation platform has the right set of capabilities to support the team. It must be able to test the hypothesis of interest to the product team. To do that one of the of the most important things required is a set of trustworthy and easily interpretable metrics to evaluate a change made to the product. In addition, it's useful if there are easy tools to manage multiple experiments and clearly communicate results from these experiments.

6.2.2 Practices, Policies and Capabilities

The second aspect deals with creating right set of practices, policies, and capabilities to encourage teams to test every change made to their product using OCEs. The following are strategies that different companies use to achieve this goal.

High Touch: Once per quarter, the LinkedIn experimentation team handpicks a few business-critical teams, prioritizes these teams, and then works closely with them on their needs. At the end of the quarter the team agrees they'll use that experiment platform going forward, and the experimentation team continues to monitor them. Over several years a data-driven culture is built. Managers and directors now rely on development teams running experiments before features launch.

The Microsoft experimentation team selects product teams to onboard based on factors indicative of the impact experimentation has on the product. The experimentation team works very closely with product teams over multiple years to advance the adoption of experimentation and its maturity over time.

The downside of the High Touch approach is the large overhead in having a deep engagement with every team, and it may become a bottleneck for scaling.

Top down buy in: It can help if there is a buy-in into experimentation by leadership and they expect every change tested in a controlled experiment. Further they can set team goals based on moving a metric in controlled experiments. This creates a culture where all ship decisions are talked about in terms of their impact on key metrics. The product teams celebrate shipping changes that improve key metrics, and equally importantly, celebrate not shipping changes that would cause a regression in key metrics. It is important that the team's key metrics are determined beforehand and agreed upon by the team. It is prudent to be cautious about preventing the gaming of metrics or over fitting metric flaws, where the metrics of interest move but are not indicative of improvement in the product. At Netflix a long-standing culture of peer review of experiment results is organized around frequent "Product Strategy" forums where results are summarized and debated amongst experimenters, product managers, and leadership teams before an experiment is "rolled out".

Negative and positive case studies: Stories about surprising negative results where a feature that is widely acclaimed as a positive causes a large regression in key metrics, or a surprising positive incident where a small change no one believed would be of consequence causes a large improvement in a metric were great drivers for cultural change. These cases drive home a humbling point that our intuition is not a good judge of the value of ideas.

There are some documented examples the best OCEs with surprising outcomes [4]. For instance, an engineer at Bing had the idea to make ad titles longer for ads with very short titles. The change was a simple and cheap, but it was not developed for many months as neither the developer nor the team had much confidence

in the idea. When it was finally tested, it caused one of the biggest increases in Bing revenue in history [51].

Safe Rollout: It is easier to get a team to adopt experimentation when it fits into their existing processes and makes them better. Some teams at Microsoft and Google began using experimentation as a way to do safe feature rollouts to all users, where an A/B test runs automatically during deployment as the feature is gradually turned on for a portion of users (Treatment) and others (Control) don't have the feature turned on. During this controlled feature rollout, the feature's impact estimate on key reliability and user-behavior metrics helped find bugs.

This method helps gain a toe hold in the feature team's development process. Over time, as the feature team started seeing value in experimentation, they looked forward to using experimentation to test more hypotheses.

Report cards and Gamification: Microsoft found that they encourage the adoption of OCEs in a set of teams by having a report card for each team that assesses their experimentation maturity level [31]. This report card gives the team a way to think about the potential of using experiments to improve the product. It gives the team a measure of its status and relative status among other teams and helps highlight key areas where they can invest to further improve.

Booking.com is experimenting with gamification in their experimentation platform where users of the platform can receive badges to encourage the adoption of good practices.

Twitter and Microsoft also use mascots, like duck [70] and HiPPO [37] to spread awareness about experimentation in their companies.

Education and support: When a company tests thousands of experiments a year, it is impossible for experimentation teams to monitor each experiment to ensure that experiment analysis is trustworthy. It is important that each team has subject matter experts to help them run experiments and ensure that they obtain reliable and trustworthy results. Educating team members on how to use OCEs to test hypotheses and how to avoid common pitfalls is critical in scaling experimentation adoption. We will discuss this important point in detail in section 7.

7. TRAINING OTHERS IN THE ORGANISATION TO SCALE EXPERIMENTATION

7.1 Problem

While the concept of an A/B test is simple, there can be complex practical issues in designing an experiment to test a particular feature and analyzing the results of the experiment. Product teams need custom support when running experiments, because they often have very specific questions that cannot be answered with a simple set of frequently answered questions.

A centralized support function does not scale very well. Central teams end up spending too much time on support and not enough on other things. Additionally, specific product domain knowledge is often required to provide support. A centralized support function requires deep knowledge of all supported products, which is often not feasible. Conversely, anyone providing support needs fundamental experimentation knowledge, which might be easier to scale. Such democratization of knowledge and expertise enables a better experimentation culture.

7.2 Common Solutions and Challenges

Across different companies, there are a few key practical challenges in spreading the expertise about OCEs that enable experimentation at scale.

- How do we set up a community to support experimenters?
- How do we incorporate them in the experiment lifecycle?
- How do we incentivize these people?
- How do we quantify their impact?
- How do we train them?
- How do we maintain quality standards?

Here are examples from several companies on how they tried to solve these challenges.

7.2.1 Yandex: "Experts on Experiment"

At Yandex, a program called "Experts on Experiment" exists to scale support. These Experts are handpicked from product teams by the central experimentation group. Any experiments must be approved by an Expert before they are allowed to ship. Experts are motivated because their product needs approval before shipping, so they voluntarily sign up to be an Expert. Their application is then reviewed by the central experimentation group. Experts are motivated by the status provided by being an Expert. They get a digital badge in internal staff systems, so their status is visible to others. There are no clear KPIs for the program. There is a checklist of minimum experience and an informal interview process involved in becoming an expert.

7.2.2 Amazon: "Weblab Bar Raisers"

Weblab is Amazon's experimentation platform. In 2013, Amazon's Personalization team piloted a "Weblab Bar Raisers" program in their local organization with the intention of raising the overall quality of experimental design, analysis, and decision making. The initial Bar Raisers were selected to be high-judgment, experienced experimenters, with an ability to teach and influence. Expectations for the role were clearly defined and documented and, after a few iterations, the program was expanded company wide. Bar Raiser review is not mandatory for all organizations; often because not enough Bar Raisers are available. Bar Raisers spend about 2–4 hours per week providing OCE support. Incentives rely on Bar Raisers buying into the mission of the program, which contributes to their personal growth and status within the company. A mentorship program, where existing Bar Raisers train new ones, exists to ensure that new Bar Raisers are brought up to speed quickly.

7.2.3 Twitter: "Experiment Shepherds"

At Twitter, the "Experiment Shepherds" program, founded three years ago by a product group including the current CTO, now has approximately 50 shepherds. Most of these are engineers with experience running experiments. There are strict entry requirements. Experiment owners implicitly opt-in for review: either pre-test or pre-launch. Shepherds have on-call duty one week a year to triage incoming requests. Incentives include feelings of responsibility for the product and acknowledgement of contribution during performance review. There are no clear impact KPIs, but qualitatively impact seems to exist. There is a structured training program consisting of one hour of classroom training per week for two months. These classes cover seven topics (e.g. dev cycle, ethics, metrics, stats 101). There are also case study-based discussions.

7.2.4 Booking.com: "Experimentation Ambassadors"

At Booking.com, the "Experimentation Ambassadors" program started about six months ago. The central experimentation organization handpicked people (~15) with experimentation experience and interest in providing support in product organizations that seemed to need the most support. Ambassadors form the first line of support with a clear escalation path and priority support from the central organization. Ambassadors are hooked into the central support ticketing system so that they are aware of other open support questions and can pick up tickets as they see fit. They are included in the experimentation organization's internal communications, to keep them aware of current developments or issues. There is a monthly meeting to discuss product needs and concerns. Incentives for Ambassadors include feeling responsible for the product, getting priority support from the central organization, and acknowledgement on their performance review. There are no clear impact or quality KPIs, but there are plans to include these as the program scales. There is no specific training for Ambassadors, but there is extensive general experiment training for all experimenters, including Ambassadors.

7.2.5 Booking.com: "Peer-Review Program"

Booking.com also has a separate "Peer-Review Program" aimed at getting people involved in providing pro-active feedback to experimenters. Anyone in the company can opt-in to the program. Every week participants are paired with a random counterpart. Currently approximately 80 people participate. Each pair picks a random experiment to review. The experiment platform includes a "give me a random experiment" button for this purpose. The platform also supports built-in commenting and threading as part of the reporting interface. Incentives to participate include making new friends, learning new things, and reward badges displayed on the platform interface. There are KPIs defined around reviews and comments. Newcomers are paired with experienced users the first few times to ensure that they are brought up to speed. A one-page guide for writing good reviews is also available [33].

7.2.6 Microsoft: Center of Excellence Model

At Microsoft, a data scientist or two from the central experimentation platform team (Analysis & Experimentation) work very closely with a product team. At first, the data scientists from the experimentation platform handle almost all support needs for the product and gain good insight into the product, business, customers, technology, and data. At the same time, the data scientists work on transferring knowledge and expertise to champions in the product team. The expectation is that over time, as more experiments are run, the product team will become more self-sufficient in running trustworthy experiments, and the person from the central experimentation platform team helps with a smaller and smaller percentage of experiments—those that are unique or have issues. The data scientists from the central team and champions from the product team usually conduct further training to educate the entire product team on best practices and processes for running experiments. The experimentation team maintains a monthly scorecard to measure the goals of each product onboarding for running trustworthy experiments at scale. These goals are set at the beginning of every year. Every six weeks, the data scientists and champions review the experimentation operations in the product where successes and failures from the past are highlighted along with a plan to address gaps and opportunities. The incentives for data scientists and champions are partially tied to the success of experimentation in their respective products.

The central experimentation team holds a weekly experiment review, where any experiment owner can share their experiment and request feedback from the data scientists. The central experimentation team also conducts a monthly Introduction to Experimentation class and Experiment Analysis lab open to everyone at Microsoft. In addition, twice a year the team hosts a meeting focused on experiments and discusses the best controlled experiments. This provides product teams an opportunity to showcase their strengths in experimentation and learn from other teams.

7.2.7 Google: Just-in-time Education Model

Google has used a variety of approaches, but one of the most successful relies heavily on just-in-time education [67]. For example, for experiment design, they have a checklist that asks experimenters a series of questions, ranging from "what is your hypothesis?" to "how will you measure success?" and "how big of a change do you need to detect?" Google has an "experiment council" of experts who review the checklists, and have found consistently that the first time through, an experimenter needs handholding. But on subsequent experiments, less handholding is needed, and the experimenter starts teaching their team members. As they become more experienced, some experimenters can become experts and perform reviews. Some teams have sufficient expertise that they can retire the entire checklist process.

For analysis, Google has an experiment review similar to Microsoft. The advantage is both just-in-time education to experimenters about interpreting experiment results and meta-analysis by experts to find the larger patterns.

8. COMPUTATION OF EXPERIMENT ANALYSIS AND METRICS

8.1 Problem

When 100s of experiments are running simultaneously on millions of users each, having an automated, reliable and efficient way to compute metrics for these experiments at scale is crucial to create a culture where OCEs are the norm. The system to compute experiment results can be viewed as a pipeline. It starts with the product collecting telemetry data points instrumented to measure user response, like clicks on a particular part of the product. The product uploads telemetry to a cloud store. This telemetry is seldom used in raw form for any analysis. Further data processing, commonly called *cooking*, joins this data with other data logs, like experiment assignment logs, and organizes it in a set of logs in standard format, called a *cooked log*. Most reporting and experiment analysis occur on top of the cooked log. For running experiments at scale, it is important to have a system for defining metrics of interest on top of these logs and actually computing metrics for each experiment running over millions of users. In addition, the system must support further ad hoc analysis of experiments so that data scientists can try different metrics and methods to find better ways of analyzing results.

There are a few key properties of a good system that help in running experiments at scale. Each part of the system must be efficient and fast to scale to 100s of experiments over millions of users each. It must be decentralized so that many people in the organization can configure and use the system to fulfill their needs. It must also have

some level of quality control to ensure that the results are trustworthy. Finally, it must be flexible enough to support the diverse needs of feature teams who are constantly working on adding new features and new telemetry, and data scientists working on new metrics and methodologies to extract insights from these experiments.

This system forms the core of experimentation analysis for any product. If done well, it empowers feature teams to run 100s of experiments smoothly and get trustworthy insights in an automated and timely manner. It helps them understand if the treatment succeeded or failed in moving the key OEC metric and gives insight into why it happened. These insights are crucial in taking next steps on an experiment: investigating a failure or investing further in successful areas. Conversely, if this system does not have the desired properties mentioned above, it often becomes a bottleneck for scaling experimentation operations and getting value from experiments.

8.2 Common Solutions and Challenges

8.2.1 Data Management and Schema

The structure and schema of cooked logs affect how data is processed in downstream data pipelines, such as metric definitions and experiment analysis. There is a clear tradeoff between reliability and flexibility. If the rules and constraints are strict, the data will be reliable and can be consumed consistently across different use cases. At the same time, having too strict constraints can slow down the implementation of the logging, and thus decelerate experimentation and product development.

Different companies have different ways of solving this issue. At Netflix, there is a single cooked log where each row is a JSON array containing all data collected. JSON structure allows flexibility and extensibility. There is a risk that the log may keep quickly changing. This must be managed by development practices to ensure that key telemetry is not lost due to a code change. A similar approach is used by MSN and Bing at Microsoft.

The bring-your-own-data approach is followed at LinkedIn, Airbnb, and Facebook. Each product team is responsible for creating data streams and metrics for each experiment unit every day. These streams follow certain guidelines that enable any experiment to use these streams to compute metrics for that experiment.

Products, like Microsoft Office, have an event-view schema, where each event is on a separate row. This format is also extensible with a more structured schema.

Another approach followed by some products is to have a fixed-set of key columns required to compute key metrics, and a property-bag column that contains all other information. This allows stability for key columns and flexibility to add new telemetry to the log.

8.2.2 Timely and Trustworthy Experiment Analysis

Many companies track hundreds of metrics in experiments to understand the impact of a new feature across multiple business units, and new metrics are added all the time. Computing metrics and providing analysis of an experiment on time is a big challenge for experimentation platforms.

As previously mentioned, in many companies, like LinkedIn, Facebook and Airbnb, the metrics framework and experimentation platform are separate, so that each product team or business unit own their metrics and is responsible for them. The experimentation platform is only responsible for the computation of metrics for experiment analysis. In other companies, like Microsoft, Google, Booking.com and Lyft, the metric computation is usually done by the experimentation team right from telemetry or cooked logs.

Individual metrics and segments can have data quality issues, delays or be computationally expensive. To resolve these issues, companies segment metrics in various ways. Having ‘tiers’ or metrics so that high-tier metrics are prioritized and thoroughly tested is a way to consume reliable experiment results. Also, if not all metrics have to be pre-computed, experimentation platforms can offer an on-demand calculation of the metrics to save computation resources.

Telemetry data from apps may have large delay getting uploaded from a section of devices. It is important to incorporate this late-arriving data in experiment analysis to avoid selection bias. Some companies like Facebook leave a placeholder for these metric values and fill it in once enough data arrives. In other companies, like LinkedIn and Microsoft, these metric values are computed with the data received at the time and then recomputed later to update the results. Usually there is a definite waiting period after which the metric value is no longer updated.

A few companies put additional steps to ensure that metrics are good quality. Some companies like LinkedIn have a committee to approve adding new metrics or modifying existing metrics to ensure metric quality. At a few companies, the metrics must be tested to ensure that they are sensitive enough to detect a meaningful difference between treatment groups. To save computational resources, the experimentation platform can require a minimum statistical power on the metrics or place metrics in specific formats. Booking.com has an automated process to detect data and metric quality issues which includes having two separate data and metric computation pipelines and process to compare the final results from both [41].

8.2.3 Metric ownership

Metrics often have an implicit or explicit owner who cares about the impact on that metric. In a large organization running 100s of experiments every day, scalable solutions ensure that these metric owners know about the experiments that move their metric, and that experiment owners know who to talk with when a particular metric moves. In many cases, it is easy to view the results of any experiment, and metric owners look for experiments that impact their metrics. Team organization structure also helps in this case. If there is a special performance team in the organization, it becomes clear to experiment owners to talk with that team when performance metrics start degrading. Some companies like Microsoft built automated systems for informing both experiment owners and metric owners when large movements are seen in a particular metric. Some teams, like performance teams, may have additional tools to search through multiple experiments to find ones that impact their metrics.

8.2.4 Supporting Exploratory and Advanced Experiment Analysis Pipelines

Very often, an experiment requires additional ad hoc analysis that cannot be supported by the regular computation pipeline. It is important that data scientists can easily conduct ad hoc analysis for experiments. Some ad hoc analyses may quickly find application in many more experiments. It is a challenge for experimentation platforms to keep up with supporting new ways of analyzing experiments while maintaining reliability and trustworthiness. While there was no common solution to solving this problem across

different companies, there are some common considerations for supporting a new analysis method:

- Is the new analysis method reliable and generalizable for all metrics and experiments?
- Is the benefit from the new method worth the additional complexity and computation?
- Which result should we rely on if the results of the experiment are different between various methods?
- How can we share the guideline so that the results are interpreted correctly?

9. DEALING WITH CLIENT BLOAT

9.1 Problem

Many experiments are run on client software (e.g., desktop and mobile). In these experiments, a new feature is coded behind a flag switched off by default. During an experiment, the client downloads a configuration, that may turn the feature flag on for that device. As more and more experiments are run over time, the configuration files that need to be sent keep growing larger and increase client bloat. This eventually starts to affect the performance of the client.

9.2 Common Solution

While it may seem that if feature F is successful it will need the flag set to ON forever, that's not the case if the experimentation system is aware of versions and which versions expect a setting for F. A key observation is that at some point when feature F is successful, it is integrated into the codebase, and from that point on, the configuration of F is NOT needed.

Here is a description of this scenario:

V10.1: Feature F is in code but not finished.

- Default (in code) = Off.
- Config: No F

V10.2 (experiment): Feature F is done.

- Default (in code) = Off
- Config: F is on/off at 50/50

If the idea fails, stop sending config for F. If the idea succeeds, Config: F=On. The key observation is that the config system must send F=On for every release that needs F as config by default, 10.2 and higher

V10.3 – Other features are evaluated.

- Config: F=On, G=On…

V10.4 – Code is cleaned.

- F=On in code. No need for F in config

Config system should stop sending F for V10.4 and higher. Every feature then has [Min version] and after cleanup [Min Version, Max version]. If we assume every release has 100 new features driven by config and 1/3 of these features are successful, the number of configuration features on the server grows at 100/3 ~ 33 per release, but only successful features should be maintained.

The number of features sent to the client is bounded by those that must be experimented and those not cleaned. Assuming three releases are needed to experiment and clean, there are 100 features in config for experiments and 100 (33 * 3 releases) maintained waiting for cleanup. This means that the total configurations are about 200, and that does not grow.

10. NETWORK INTERACTIONS

10.1 Problem

Network interactions are a significant concern in A/B testing. Traditional A/B test assume a stable user treatment value (SUTVA) to accurately analyze the treatment effect. SUTVA implies that the response of an experiment unit (user) is independent of the response of another experiment unit under treatment [73]. A network interaction can occur when a user's behavior is influenced by another user's, so that users in the control group are influenced by actions taken by members in the treatment group. As a result, the control group is only a control group in name and no longer reflect outcomes that would be observed if the treatment did not exist. If you ignore network interactions, you get a biased estimate of the treatment effect.

10.2 Common Solutions and Challenges

These network interactions are an inherent outcome of the products and scenarios where changes are being tested. There does not seem to be one single method that can mitigate the impact of network interactions on the accuracy of the estimated treatment effect. Here are some common cases and the methods to deal with them.

10.2.1 Producer and Consumer Model

At LinkedIn, there is a meaningful producer/consumer distinction between user roles for a feature. For instance, there are producers and consumers of the hashtags feature for the main feed on LinkedIn. In these cases, LinkedIn typically uses *two-sided randomization*. Two orthogonal experiments are run together: one controlling the production experience and one controlling the consumption experience. For the hashtags example, this implies that the production experiment allows users in treatment to add hashtags to their posts, and the consumption experiment allows users in treatment to see hashtags on their feed. The production experience starts at a low ramp percentage with consumption one at a high percentage, and then gradually ramping the production experience.

If we do a simple A/B test lumping both features together, then things go wrong: The producer effect is underestimated because there are too few potential consumers. For our example, if a user in treatment in the production experiment can post hashtags but not everybody can see them, then the user is likely to engage less with the platform. The consumer effect is underestimated because there are too few potential producers. Being able to see hashtags may make users more engaged, but not if too few people (i.e. only treated members) use them. Using two sided randomization helps: when 95% of consumers can see the produced content, then the effect of producers (say at 50% ramp) is more accurate; when 95% of producers are "enabled," then the consumer test (say 50% ramp) is more accurate.

This method may not account for competition effects between producers, in which case we typically use a 95% ramp over 50% ramp if enough power is available. Further, it may not be possible to disentangle consumption from production in a feature. For instance, if a user mentions another user using '@ mention' feature, then the consumer of the feature must be notified about being mentioned.

10.2.2 Known Influence Network Model

In many products at LinkedIn and Facebook, the network over which users can influence each other is known. This information is helpful for designing better controlled experiments.

LinkedIn typically uses its egoClusters method, creating about 200,000 ego-networks, comprised of an "ego" (the individual whose metrics are measured) and "alters," who receive treatments but whose metrics are not of interest. Clusters are designed to have egos representative of LinkedIn users and their networks, and treatment is allocated as follows: in all clusters, egos are treated. In "treated" clusters, all alters are treated. In control clusters, all alters remain in control. A simple two-sample t-test between egos of treated clusters and egos of control clusters gives the approximate first-order effect of having all their connections treated versus none.

Facebook and Google employ similar cluster based randomization techniques [20, 26]. These designs are the subject of recent academic papers [9].

10.2.3 One-to-One Communication

When the feature being tested is one-to-one communication, LinkedIn typically uses model-based approaches when analyzing one-to-one messaging experiments, counting messages explicitly according to four categories: those that stay within the treatment group, those that stay within the control group, and those that cross (one way or the other). The total number of messages of these categories are contrasted with the help of a model and permutation testing to measure the impact of network interactions.

At Skype, some experiments related to call quality are randomized at the call level, where each call has an equal probability of being treatment or control. Note that a single user may make multiple calls during the experiment. This approach does not account for within-user effect from a treatment but tends to have much greater statistical power for detecting the treatment effect on the call metrics.

10.2.4 Market Effects

In a two-sided marketplace, different users' behavior is correlated with each other due to a demand-and-supply curve. If we look at a ride service, when a driver is matched to a passenger, it lowers the probability that other drivers in vicinity are matched. Simple randomization of passengers or drivers into Treatment and Control groups causes changes in market conditions, therefore biases the estimated Treatment effect. To reduce the network interactions between users, Lyft conducts cluster sampling by randomizing across spatial regions or time intervals of varying size, ensuring similarity in market conditions between variants. The coarser the experimental units are, the less interference bias persists, although it comes with the cost of increased variance in the estimate [29]. Uber has tried introducing the treatment to a random set of markets and have a synthetic control to predict the counterfactual [1, 34].

Similar market effects also affect online ads. In this hypothetical example, assume that all budget for a set of advertisers is being spent. For the experiment, the treatment increases ad load from these advertisers therefore increasing ad consumption. In this experiment, you would observe that revenue in the treatment group goes up. But the treatment group is stealing budget from the control group, and there will be no increase in revenue when the treatment ships to all users.

One way to prevent budget stealing is to split the ad budget of all ad providers in proportion to the percentage of user traffic exposed to the treatment and control groups. While this addresses the problem of budget stealing, it does not help us understand if the treatment will cause an increase in revenue. Higher use of budgets not being entirely spent or an increase in budget from advertisers spending their entire budget may be a better indicator of increase in revenue.

10.2.5 Multiple Identities for the Same Person

Similar statistical issues arise when the same user has several accounts or cookies. Instead of spillover occurring from one user to another, it may occur from one account to another, within the same user. A natural level of randomization is user. However, this requires knowing which accounts belong to the same user. If this is unknown or imperfectly known, randomization at the account-level may be the only alternative. Account-level randomization generally tends to suffer from attenuation bias. Studies in Facebook have indicated that cookie level randomization can underestimate person level effects by a factor of 2 or 3 [15]. Attenuation bias is also one of the main pitfalls in running long-term experiments because the chances of within-user spillover increases with time [23].

11. INTERACTIONS BETWEEN MULTIPLE EXPERIMENTS

11.1 Problem

If there are non-independent treatment effects in two experiments, then those experiments are said to be interacting:

$$ATE(T_1) + ATE(T_2) \neq ATE(T_1 T_2)$$

A textbook example of interaction between two experiments is where the treatment in the first experiment changes the foreground color to blue and the treatment in the second experiment changes the background color to blue. In this example let us assume that there are positives for each experiment in isolation, but the impact of both treatments is catastrophic. A user who experiences both treatments at the same time sees a blue screen.

In products where 100s of experiments run concurrently this can be a serious issue. Ideally you want to prevent contamination where the treatment effect measured in one experiment may become biased because that experiment interacts with another experiment. At the same time, you need to make a joint ship decision for interacting experiments. As in the case of the text book example above, individually both treatments are good ship candidates but jointly you can only ship one.

11.2 Common Solutions and Challenges

From our experience, it is rare that two interacting experiments cause enough contamination that it changes the ship decision. Most products are well architected and small teams work independently of most other teams working on different areas of the product. The chances of interaction between two experiments are highest when both experiments are being run by the same sub team who are changing the same part of the product. To prevent interaction between these types of experiments, the Microsoft and Google experimentation platforms have the concept of *numberlines* or *layers* [46, 68]. Experiments that run on the same numberline or layer are guaranteed to get an exclusive random sample of the user population, so no user is exposed to two experiments being run concurrently in the same layer or numberline. This limits the

number of users who can be part of an experiment. If the first experiment is exposed to half of all users, then the second experiment cannot be exposed to more than remaining half of the user base. Small teams manage a group of numberlines or layers. Based on their understanding of the treatments in different experiments, the teams can decide whether to run the experiments in the same numberline/layer.

To detect interactions between two experiments running in two different layers, Microsoft runs a daily job that tests each pair of experiments for additivity of their treatment effects: $\mu(T_1C_2) - \mu(C_1C_2) \neq \mu(T_1T_2) - \mu(C_1T_2)$.

It is rare to detect interactions between two experiments as experiment owners already try to isolate experiments that may conflict by running them on the same numberline or layer.

To address the problem of joint decision making, you can run both experiments on different numberlines or layers—if we know that the combination of two experiments cannot lead to a catastrophic result. In this case, you can analyze the factorial combination of both experiments to understand the effect of treatment from each experiment individually and the effect of treatments from both experiments.

12. CONCLUSION

This is the first paper that brings together the top practical challenges in running OCEs at scale from thirty-four experts in thirteen different organizations with experience in testing more than one hundred thousand treatments last year alone. These challenges broadly fall into four categories: analysis of experiments, culture and engineering, deviations from traditional A/B tests, and data quality. In Sections 3-5, we discussed the problem that while most experiments run for a short period of time, we want to estimate the long term impact of a treatment and define an overall evaluation criteria (OEC) to make ship decisions for all experiments in a consistent and objective manner while taking into account the heterogenous treatment effects across different product and user segments. In sections 6-9, we discussed the importance of culture and engineering systems in running OCEs at scale. We discussed common challenges and approaches in making OCEs the default method for testing any product change and scaling OCE expertise across the company. We also discussed some common challenges and solutions for computation of experiment analysis and metrics, and client bloat due to configurations from a large number of OCEs. In Sections 10 and 11, we discussed problems and challenges arising from some common deviations from traditional OCEs due to inherent network interactions in different product scenarios and interactions between experiments. There are many more issues of great importance like privacy, fairness and ethics that are handled in each company individually and often form the underlying subtext of the analysis methods and best practices including expert supervision and review described in this paper. We hope to discuss these topics in more detail in future summits/meetups. We hope this paper sparks further research and cooperation in academia and industry on these problems.

13. ACKNOWLEDGMENTS

We would like to thank Stacie Vu from LinkedIn and Michele Zunker from Microsoft for their help in making this summit possible, and Cherie Woodward from Docforce for helping edit the paper. We would also like to thank our colleague Widad Machmouchi and John Langford for their feedback.

14. ADDITIONAL AUTHOR INFORMATION

Somit Gupta (somit.gupta@microsoft.com), Ronny Kohavi (ronnyk@microsoft.com), Diane Tang (diane@google.com), Ya Xu (yaxu@linkedin.com), Reid Andersen (reid.andersen@airbnb.com), Eytan Bakshy (ebakshy@fb.com), Niall Cardin (niallc@google.com), Sumitha Chandran (schandran@lyft.com), Nanyu Chen (nchen@linkedin.com), Dominic Coey (coey@fb.com), Mike Curtis (mikecurtis@google.com), Alex Deng (alexdeng@microsoft.com), Weitao Duan (wduan@linkedin.com), Peter Forbes (pforbes@netflix.com), Brian Frasca (brianfra@microsoft.com), Tommy Guy (riguy@microsoft.com), Guido W. Imbens (imbens@stanford.edu), Guillaume Saint Jacques (gsaintjacques@linkedin.com), Pranav Kantawala (pranav@google.com), Ilya Katsev (bromozel@yandex-team.ru), Moshe Katzwer (mkatzwer@uber.com), Mikael Konutgan (kmikael@fb.com), Elena Kunakova (ensuetina@yandex-team.ru), Minyong Lee (minyong.lee@airbnb.com), MJ Lee (mjlee@lyft.com), Joseph Liu (josephl@twitter.com), James McQueen (jmcq@amazon.com), Amir Najmi (amir@google.com), Brent Smith (smithbr@amazon.com), Vivek Trehan (vivek@uber.com), Lukas Vermeer (lukas.vermeer@booking.com, Booking.com), Toby Walker (towalker@microsoft.com), Jeffrey Wong (jeffreyw@netflix.com), Igor Yashkov (excel@yandex-team.ru)

15. REFERENCES

[1] [Uber Marketplace] Marketplace Experimentation -- Vivek Trehan - YouTube: 2018. *https://www.youtube.com/watch?v=IR000RqN7pw*. Accessed: 2019-02-05.

[2] 25 Cognitive Biases Home Page | 25 Cognitive Biases - "The Psychology of Human Misjudgment" *http://25cognitivebiases.com/*. Accessed: 2019-02-08.

[3] A/B Testing at Scale Tutorial Strata 2018: *https://exp-platform.com/2018StrataABtutorial/*. Accessed: 2019-02-05.

[4] AdvancedTopic_BestControlledExperiments.docx - Microsoft Word Online: *https://onedrive.live.com/view.aspx?resid=8612090E610871E4!323967&ithint=file%2Cdocx&app=Word&authkey=!APyJuF_t0dOFj_M*. Accessed: 2019-02-05.

[5] AdvancedTopic_OEC.docx - Microsoft Word Online: *https://onedrive.live.com/view.aspx?resid=8612090E610871E4!282179&ithint=file%2Cdocx&app=Word&authkey=!ANFGOBrhVt91ODk*. Accessed: 2019-02-05.

[6] Agarwal, A. et al. 2016. Making Contextual Decisions with Low Technical Debt. (Jun. 2016).

[7] Amazon.com case study - 2018 update | Smart Insights: 2018. *https://www.smartinsights.com/digital-marketing-strategy/online-business-revenue-models/amazon-case-study/*. Accessed: 2019-02-04.

[8] Athey, S. et al. 2016. Estimating Treatment Effects using Multiple Surrogates: The Role of the Surrogate Score and the Surrogate Index. *arXiv*. (2016). DOI:https://doi.org/10.1039/C002690P.

[9] Athey, S. et al. 2018. Exact p -Values for Network

Interference. *Journal of the American Statistical Association*. 113, 521 (Jan. 2018), 230–240. DOI:https://doi.org/10.1080/01621459.2016.1241178.

[10] Bakshy, E. et al. 2014. Designing and deploying online field experiments. *Proceedings of the 23rd international conference on World wide web - WWW '14* (New York, New York, USA, 2014), 283–292.

[11] Bloodletting – ExP Platform: *https://exp-platform.com/bloodletting/*. Accessed: 2019-02-05.

[12] Building an Intelligent Experimentation Platform with Uber Engineering: *https://eng.uber.com/experimentation-platform/*. Accessed: 2019-02-04.

[13] Chen, N. et al. 2018. Automatic Detection and Diagnosis of Biased Online Experiments. *arXiv preprint arXiv:1808.00114*. (2018).

[14] Chernozhukov, V. et al. 2016. *Double machine learning for treatment and causal parameters*.

[15] Coey, D. and Bailey, M. 2016. People and Cookies: Imperfect Treatment Assignment in Online Experiments. *Proceedings of the 25th International Conference on World Wide Web - WWW '16* (2016), 1103–1111.

[16] Crook, T. et al. 2009. Seven pitfalls to avoid when running controlled experiments on the web. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09* (New York, New York, USA, 2009), 1105.

[17] Dan McKinley :: Testing to Cull the Living Flower: *https://mcfunley.com/testing-to-cull-the-living-flower*. Accessed: 2019-02-10.

[18] Deng, A. et al. 2013. Improving the sensitivity of online controlled experiments by utilizing pre-experiment data. *Proceedings of the sixth ACM international conference on Web search and data mining - WSDM '13* (New York, New York, USA, 2013), 123.

[19] Deng, A. and Shi, X. 2016. Data-Driven Metric Development for Online Controlled Experiments. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16* (2016), 77–86.

[20] Designing A/B tests in a collaboration network: 2018. *http://www.unofficialgoogledatascience.com/2018/01/designing-ab-tests-in-collaboration.html*. Accessed: 2019-02-12.

[21] Devore, J.L. and Berk, K.N. 2012. *Modern Mathematical Statistics with Applications*. Springer.

[22] Dmitriev, P. et al. 2017. A Dirty Dozen: Twelve Common Metric Interpretation Pitfalls in Online Controlled Experiments. *Proceedings of the 23rd ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '17* (Halifax, Nova Scotia, Canada, 2017).

[23] Dmitriev, P. et al. 2016. Pitfalls of long-term online controlled experiments. *2016 IEEE International Conference on Big Data (Big Data)* (Washington, DC, USA, Dec. 2016), 1367–1376.

[24] Dmitriev, P. and Wu, X. 2016. Measuring Metrics. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management - CIKM '16* (2016), 429–437.

[25] Drutsa, A. et al. 2017. Using the Delay in a Treatment Effect to Improve Sensitivity and Preserve Directionality of Engagement Metrics in A/B Experiments. *Proceedings of the 26th International Conference on World Wide Web - WWW '17*. (2017), 1301–1310. DOI:https://doi.org/10.1145/3038912.3052664.

[26] Eckles, D. et al. 2017. Design and analysis of experiments in networks: Reducing bias from interference. *Journal of Causal Inference*. 5, 1 (2017), 23. DOI:https://doi.org/10.1515/jci-2015-0021.

[27] Experimentation – Booking.com Data Science: *https://booking.ai/tagged/experimentation*. Accessed: 2019-02-04.

[28] Experimentation and Testing: A Primer - Occam's Razor by Avinash Kaushik: 2006. *https://www.kaushik.net/avinash/experimentation-and-testing-a-primer/*. Accessed: 2019-02-10.

[29] Experimentation in a Ridesharing Marketplace - Lyft Engineering: *https://eng.lyft.com/experimentation-in-a-ridesharing-marketplace-b39db027a66e*. Accessed: 2019-02-04.

[30] Experiments at Airbnb – Airbnb Engineering & Data Science – Medium: *https://medium.com/airbnb-engineering/experiments-at-airbnb-e2db3abf39e7*. Accessed: 2019-02-04.

[31] Fabijan, A. et al. 2018. Experimentation growth: Evolving trustworthy A/B testing capabilities in online software companies. *Journal of Software: Evolution and Process*. (Nov. 2018), e2113. DOI:https://doi.org/10.1002/smr.2113.

[32] Fabijan, A. et al. 2018. Online Controlled Experimentation at Scale: An Empirical Survey on the Current State of A/B Testing. *Proceedings of the 2018 44rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (Prague, Czechia., 2018).

[33] Fabijan, A. et al. 2019. Three Key Checklists and Remedies for Trustworthy Analysis of Online Controlled Experiments at Scale. *to appear in the proceedings of 2019 IEEE/ACM 39th International Conference on Software Engineering (ICSE) Software Engineering in Practice (SEIP)* (Montreal, Canada, 2019).

[34] Fosset Jeff, Gilchrist Duncan, L.M. 2018. Using Experiments to Launch New Products. *Harvard Business Review*.

[35] Groysberg, B. et al. 2018. The Leader's Guide to Corporate Culture. *Harvard Business Review*.

[36] Hassan, A. et al. 2013. Beyond clicks. *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13* (New York, New York, USA, 2013), 2019–2028.

[37] HiPPO – ExP Platform: *https://exp-platform.com/hippo/*. Accessed: 2019-02-08.

[38] Hohnhold, H. et al. 2015. Focusing on the Long-term. *Proceedings of the 21th ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining - KDD '15 (New York, New York, USA, 2015), 1849–1858.

[39] Imbens, G.W. and Rubin, D.B. 2015. *Causal inference: For statistics, social, and biomedical sciences an introduction*. Cambridge University Press.

[40] It's All A/Bout Testing – Netflix TechBlog – Medium: 2016. *https://medium.com/netflix-techblog/its-all-a-bout-testing-the-netflix-experimentation-platform-4e1ca458c15*. Accessed: 2019-02-04.

[41] Kaufman, R.L. et al. 2017. Democratizing online controlled experiments at Booking. com. *arXiv preprint arXiv:1710.08217*. (2017), 1–7.

[42] Kharitonov, E. et al. 2017. Learning Sensitive Combinations of A/B Test Metrics. *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining - WSDM '17*. (2017), 651–659. DOI:https://doi.org/10.1145/3018661.3018708.

[43] Kluck, T. and Vermeer, L. 2017. Leaky Abstraction In Online Experimentation Platforms: A Conceptual Framework To Categorize Common Challenges. *arXiv*. (Oct. 2017).

[44] Kohavi, R. et al. 2009. Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery*. 18, 1 (Feb. 2009), 140–181. DOI:https://doi.org/10.1007/s10618-008-0114-1.

[45] Kohavi, R. 2014. Lessons from Running Thousands of A/B Tests. *The Conference on Digital Experimentation (CODE@MIT) 10th - 11th of October 2014* (2014).

[46] Kohavi, R. et al. 2013. Online controlled experiments at large scale. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13* (Chicago, Illinois, USA, 2013), 1168.

[47] Kohavi, R. et al. 2009. Online experimentation at Microsoft. *Third Workshop on Data Mining Case Studies and Practice Prize*. (2009), 1–11. DOI:https://doi.org/10.1002/adfm.200801473.

[48] Kohavi, R. et al. 2014. Seven rules of thumb for web site experimenters. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14* (New York, USA, 2014), 1857–1866.

[49] Kohavi, R. et al. 2012. Trustworthy online controlled experiments: Five Puzzling Outcomes Explained. *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '12* (New York, New York, USA, 2012), 786.

[50] Kohavi, R. and Longbotham, R. 2011. Unexpected results in online controlled experiments. *ACM SIGKDD Explorations Newsletter*. 12, 2 (Mar. 2011), 31. DOI:https://doi.org/10.1145/1964897.1964905.

[51] Kohavi, R. and Thomke, S. 2017. The Surprising Power of Online Experiments. *Harvard Business Review*.

[52] Letham, B. et al. 2019. Constrained Bayesian Optimization with Noisy Experiments. *Bayesian Analysis*. 14, 2 (Jun. 2019), 495–519. DOI:https://doi.org/10.1214/18-BA1110.

[53] Li, L. et al. 2010. A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th international conference on World wide web - WWW '10* (New York, New York, USA, 2010), 661.

[54] Machmouchi, W. et al. 2017. Beyond Success Rate. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management - CIKM '17* (New York, New York, USA, 2017), 757–765.

[55] Machmouchi, W. and Buscher, G. 2016. Principles for the Design of Online A/B Metrics. *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval - SIGIR '16* (New York, New York, USA, 2016), 589–590.

[56] McFarland, C. 2012. *Experiment!: Website conversion rate optimization with A/B and multivariate testing*. New Riders.

[57] Mehrotra, R. et al. 2017. User Interaction Sequences for Search Satisfaction Prediction. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17*. (2017), 165–174. DOI:https://doi.org/10.1145/3077136.3080833.

[58] Microsoft Experimentation Platform: *http://exp-platform.com*.

[59] Poyarkov, A. et al. 2016. Boosted Decision Tree Regression Adjustment for Variance Reduction in Online Controlled Experiments. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*. (2016), 235–244. DOI:https://doi.org/10.1145/2939672.2939688.

[60] Quicken Loan's Regis Hadiaris on multivariate testing - Biznology: *https://biznology.com/2008/12/multivariate_testing_in_action/*. Accessed: 2019-02-10.

[61] Ramblings on Experimentation Pitfalls, Part 1. – Lyft Engineering: *https://eng.lyft.com/ramblings-on-experimentation-pitfalls-dd554ff87c0e*. Accessed: 2019-02-04.

[62] Rodden, K. et al. 2010. Measuring the user experience on a large scale. *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10* (New York, New York, USA, 2010), 2395.

[63] Schurman Eric, B.J. 2009. Performance Related Changes and their User Impact. *Velocity* (2009).

[64] Schurman Eric, B.J. 2009. The User and Business Impact of Server Delays, Additional Bytes, and HTTP Chunking in Web Search: Velocity 2009 - O'Reilly Conferences, June 22 - 24, 2009 - San Jose, CA. *Velocity* (2009).

[65] Semmelweis Reflex – ExP Platform: *https://exp-platform.com/semmelweis-reflex/*. Accessed: 2019-02-05.

[66] Statistics for Bioinformatics: 2008. *https://www.stat.berkeley.edu/~mgoldman/Section0402.pdf*. Accessed: 2019-02-05.

[67] Tang, D. et al. 2010. Overlapping experiment infrastructure. *Proceedings of the 16th ACM SIGKDD*

international conference on Knowledge discovery and data mining - KDD '10 (2010), 17.

[68] Tang, D. et al. 2010. Overlapping experiment infrastructure. *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10* (New York, New York, USA, 2010), 17.

[69] Tibshirani, R. et al. Sparsity and smoothness via fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*. 67, 1 (Feb.), 91–108. DOI:https://doi.org/10.1111/j.1467-9868.2005.00490.x.

[70] Twitter experimentation: technical overview: 2015. *https://blog.twitter.com/engineering/en_us/a/2015/twitter-experimentation-technical-overview.html*. Accessed: 2019-04-02.

[71] Two-sided market: *https://en.wikipedia.org/wiki/Two-sided_market*. Accessed: 2019-10-02.

[72] Universally unique identifier: *https://en.wikipedia.org/wiki/Universally_unique_identifier*. Accessed: 2019-08-02.

[73] Violations of SUTVA | Social Science Statistics Blog: 2009. *https://blogs.iq.harvard.edu/violations_of_s*. Accessed: 2019-02-04.

[74] Wager, S. and Athey, S. 2018. Estimation and Inference of Heterogeneous Treatment Effects using Random Forests. *Journal of the American Statistical Association*. 113, 523 (Jul. 2018), 1228–1242. DOI:https://doi.org/10.1080/01621459.2017.1319839.

[75] Xie, H. and Aurisset, J. 2016. Improving the Sensitivity of Online Controlled Experiments. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*. (2016), 645–654. DOI:https://doi.org/10.1145/2939672.2939733.

[76] Xu, Y. et al. 2015. From Infrastructure to Culture: A/B Testing Challenges in Large Scale Social Networks. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA, 2015), 2227–2236.

[77] Zhao, Y. et al. 2012. Estimating Individualized Treatment Rules Using Outcome Weighted Learning. *Journal of the American Statistical Association*. 107, 499 (Sep. 2012), 1106–1118. DOI:https://doi.org/10.1080/01621459.2012.695674.