Diagnosing Sample Ratio Mismatch in Online Controlled Experiments: A Taxonomy and Rules of Thumb for Practitioners

Aleksander Fabijan, Jayant Gupchup, Somit Gupta, Jeff Omhover, Wen Qin
Microsoft Corporation
Redmond, USA
{alfabija, jayagup, sogupta, jeomhove, weqin}@microsoft.com

Lukas Vermeer
Booking.com
Amsterdam, Netherlands
lukas.vermeer@booking.com

Pavel Dmitriev
Outreach.io
Seattle, USA
pavel.dmitriev@outreach.io

ABSTRACT

Accurately learning what delivers value to customers is difficult. Online Controlled Experiments (OCEs), aka A/B tests, are becoming a standard operating procedure in software companies to address this challenge as they can detect small causal changes in user behavior due to product modifications (e.g. new features). However, like any data analysis method, OCEs are sensitive to trustworthiness and data quality issues which, if go unaddressed or unnoticed, may result in making wrong decisions. One of the most useful indicators of a variety of data quality issues is a Sample Ratio Mismatch (SRM) – the situation when the observed sample ratio in the experiment is different from the expected. Just like fever is a symptom for multiple types of illness, an SRM is a symptom for a variety of data quality issues. While a simple statistical check is used to detect an SRM, correctly identifying the root cause and preventing it from happening in the future is often extremely challenging and time consuming. Ignoring the SRM without knowing the root cause may result in a bad product modification appearing to be good and getting shipped to users, or vice versa. The goal of this paper is to make diagnosing, fixing, and preventing SRMs easier. Based on our experience of running OCEs in four different software companies in over 25 different products used by hundreds of millions of users worldwide, we have derived a taxonomy for different types of SRMs. We share examples, detection guidelines, and best practices for preventing SRMs of each type. We hope that the lessons and practical tips we describe in this paper will speed up SRM investigations and prevent some of them. Ultimately, this should lead to improved decision making based on trustworthy experiment analysis.

CCS CONCEPTS

• General and reference → Experimentation • General and reference → Empirical studies • Computing methodologies → Causal reasoning and diagnostics

Keywords: A/B Testing, Online Controlled Experiments, Sample Ratio Mismatch, SRM

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
KDD '19, August 4–8, 2019, Anchorage, AK, USA
© 2019 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330722

ACM Reference format:

Aleksander Fabijan, Jayant Gupchup, Somit Gupta, Jeff Omhover, Wen Qin, Lukas Vermeer, and Pavel Dmitriev. 2019. Diagnosing Sample Ratio Mismatch in Online Controlled Experiments: A Taxonomy and Rules of Thumb for Practitioners. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'19), August 4 - 8, 2019, Anchorage, Alaska.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330722.

1. INTRODUCTION

Online Controlled Experiments (OCEs), A/B tests, or simply experiments are becoming a standard operating procedure in data-driven software companies [1, 11, 21, 22, 24]. In the simplest OCE, two variants of a product are simultaneously exposed to two randomly selected groups of users. Correctly executed experiments can accurately measure the effect of changes to the product on user behavior and key metrics, increase the quality of a product, and align the organization around a unifying goal [12, 22]. These benefits, however, can only be achieved if experiments are executed and analyzed correctly. For that, data scientists meticulously examine OCEs to rule out any data quality issues that could invalidate the results of their experiments [33]. Consider for example the following OCE that ran at Microsoft. A product team at MSN increased the number of rotating cards on the carousel from 12 to 16 (see Figure 1).

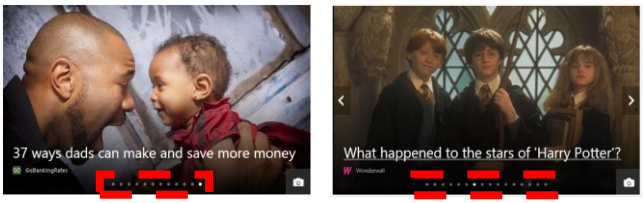

Figure 1. The MSN Carousel Experiment.

The expectation before the experiment was to see an increase in the number of times that users click and engage with the carousel [8, 29]. This experiment had enough statistical power to detect very small changes, user interaction telemetry was correctly logged and collected, and the platform on which the experiment ran produced trustworthy analysis. Contrary to what the team was hoping for based on the learnings from related experiments that showed an increase in clicks, this experiment revealed a significant decrease in engagement with the cards. Users seeing more cards appeared to click less. Based on this result it seemed like a bad idea to ship this change. What happened?

One clue for this outcome was a data quality warning firing in the experimentation platform [28]. This warning indicated an

unexpected proportion of users in the experiment variants. In particular, the treatment group which was exposed to the new 16 card carousel had fewer users in the analysis compared to what was configured and expected based on the experiment set-up. The deviations from the configured split were highly statistically significant so unlikely to happen just due to chance. [8, 19]. This difference is commonly known as a Sample Ratio Mismatch (SRM) and prominently surfacing it to experimenters is critical to prevent them from making inaccurate conclusions [17]. After resolving the issue that was causing the SRM in experiment above (see section 5.3 for details), it was found that the change was in fact positive and significantly increased the engagement with the product.

An SRM is one of the most useful indicators of many common and severe data quality issues in Online Controlled Experiments [4, 5, 13, 18]. While testing whether an experiment exhibits an SRM is straightforward, investigating the root cause can be very challenging even for experienced analysts [9]. For example, while the SRM warning in the experiment above indicated that something went wrong with the experiment, it did not reveal why. The 'why' was determined through a deep-dive analysis by experienced analysts.

Experimentation trustworthiness issues are like cancer for a product. If left ignored, these issues multiply quickly. SRM is a good indicator of this problem. While some types of SRMs are common (e.g. SRMs due to log processing are highlighted in [4, 35]), SRM conditions can happen anywhere in the experiment lifecycle. Are the experiment variants correctly logging telemetry [3]? Are experimenters or users interfering with the experiment? These are just a few of the many possible scenarios that analysts in our case study companies evaluate during their experiment analysis process. As illustrated in section 4, searching for the root cause can take weeks, sometimes even months. During this time, the experiment owners are paralyzed as the decision on whether the new feature is good or bad cannot be conclusively determined.

In this paper, we describe the common causes for an SRM, discuss analysis steps to effectively diagnose each of them and share best practices for preventing those that can be prevented. We base our findings on case study research [31] at Booking.com, Microsoft, Outreach.io, and Online Dialogue. Our main contribution is threefold:

1. We provide a taxonomy of common SRM conditions.
2. We provide real example experiments that exhibited SRM conditions for each of the taxonomy branches.
3. We share best practices and rules of thumb for detecting and preventing SRM conditions.

While several examples of SRMs were discussed before in the literature, to our knowledge this is the first large-scale systematic study of common SRM types.

2. BACKGROUND

2.1 Online Controlled Experiments

Online Controlled Experiments, A/B test, or simply experiments are widely used by data-driven companies to evaluate the impact of software changes (e.g. new features) [11]. OCEs such as the one that we introduced in the introduction are used for evaluating changes on web sites [15, 16], mobile and desktop apps, gaming consoles, social networks [34], operating systems [21] and other platforms. 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 a new feature added, say, feature X. User interactions with the system are recorded and from that, metrics are computed. If the experiment was designed and executed correctly, the only thing consistently different between the two variants is the feature X. External factors such as seasonality, impact of other feature launches, moves by competition, etc. are distributed evenly between control and treatment. Hence, any difference in metrics between the two groups can be attributed to either the feature X or random chance. The latter hypothesis is ruled out using statistical tests such as a t-test [6]. This establishes a causal relationship between the change made to the product and changes in user behavior, which is the key reason for widespread use of controlled experiments [11].

2.2 Online Controlled Experiment Steps

Every online controlled experiment consists of several steps that need to be executed correctly for OCE to be valid. In our previous research [14], we described the four key steps in the experiment process: (1) *Experiment Assignment* – the initial stage where users are split based on some unique identifier into groups and assigned one of the product variations. (2) *Experiment Execution* – the stage where users actually receive a variant based on the earlier assignment (e.g. the correct configuration is downloaded to a client), the variant is applied, and its usage is being logged. (3) *Experiment Log Processing* – The stage where the telemetry generated in the second stage is collected from the client (e.g. uploaded from browser storage to some cloud storage) and processed (e.g. joined with server logs). (4) *Experiment Analysis* – The final stage where processed logs are analyzed by e.g. filtering down to users impacted by the change. The characterization into the aforementioned four stages can be derived also by studying the architectural and infrastructural descriptions in papers on experimentation published by other researchers (see [2, 33, 34]). As we will explain later, there are distinct types of SRMs that arise in each of the experiment steps presented in this section.

2.3 Data Quality

One of the most critical components of experimentation is data quality. Practitioners at Microsoft [7, 13, 20], Booking.com [17], LinkedIn [4] and other companies frequently report on experiments that are at risk of being incorrectly analyzed due to various issues with data. One of the most important symptoms of data quality issues is the Sample Ratio Mismatch.

2.3.1 Sample Ratio Mismatch

Sample Ratio Mismatch, or simply 'SRM' is a data quality check that indicates a significant difference between expected proportions of users among experiment variants (e.g. configured before the experiment started) and the actual proportions of users observed at the end of the experiment. For example, if a 50/50 split is expected between two experiment variants, the ratio between the number of users exposed to each of the groups at the end of the experiment is expected to be close to 1. While there are many data quality issues that could decrease the validity and significance of a controlled experiment, Sample Ratio Mismatch in most cases completely invalidates experiment results. For example, a ratio of 50.2/49.8 (821,588 versus 815,482 users) diverges enough from an expected 50/50 ratio that the probability that it happened by chance is less than 1 in 500k. To detect an SRM, a chi-square test [30] can be used. If the test determines an unlikely difference between the configured and observed numbers, analysts must investigate the root cause.

2.3.2 *Diagnosing Sample Ratio Mismatches*

Several specific SRM root causes were extensively studied by researchers and practitioners. For example, Chen et al. at LinkedIn [4] exposed SRMs that happen during a triggered analysis of an experiment and shared the lessons learned while building a toolkit for diagnosing this type of SRM. They revealed that about 10% of triggered analysis at LinkedIn have an SRM. A triggered analysis is a type of experiment evaluation where only the users that were affected by the change (those in the treatment group that were exposed to the treatment) and the users that would have been affected if they were exposed to it (counterfactual - the users in the control group) are included in the analysis process. Here, a wrong counterfactual or triggered condition can cause an SRM. Next, Zhao et al. at Yahoo [35] recognized SRM cases due to incorrect treatment assignment, lost telemetry and issues in the design of the trial. In our prior research at Microsoft [10], we shared example SRMs that happened in the data processing (cooking) phase like the bot example presented in section 5.3, as well as in the analysis phase due to wrong triggering or filtering criteria.

Lessons learned from expanding experimentation to tens of thousands of experiments yearly and to devices that span beyond web sites and mobile applications taught us to seek for root causes beyond the obvious suspects presented in the prior literature. For example, SRMs can also be caused by 'power users' modifying product telemetry, by experimenters that manipulate experiment variants, by systems that deploy experiment variants, and so on. This can be very challenging and costly, hence the need to study the common SRM scenarios and provide a topology to improve our capacity to identify root causes.

3. RESEARCH METHOD

To conduct this research, we collected and analyzed both qualitative and quantitative data from four case companies - Microsoft, Booking.com, Outreach.io, and Online Dialog. We used a mixed method approach: a combination of qualitative and quantitative methods is commonly used in case study research [31] to study a problem from different angles and to synthesize the available data [26]. The goal of our study was to learn what the common SRM types are, how analysts diagnose them, and whether some of the SRMs can be prevented. To learn about these, we conducted four types of data collection and analysis efforts. First, we reviewed recent and related literature for reported SRM scenarios. Second, we collected and analyzed qualitative data by examining internal documentation available to the authors of this paper. Third, we conducted 14 interviews with experienced analysts and developers working at the case companies based on a set of semi-directive questions. The questionnaire itself is available at [32]. Finally, we performed a quantitative analysis of the historical data from previous controlled experiments that ran at our case companies to get an insight on what common and severe SRM cases really are. In total, the sample size of our historical data is over 10000 OCEs.

4. SRM LEARNINGS AND INSIGHTS

In this section, we provide several learnings obtained during this study. Note that this is not intended to be a comprehensive list, but rather a short discussion confirming the importance of SRMs.

4.1 SRMs are a Common Data Quality Issue

Recent research contributions from large scale companies such as LinkedIn [4] and Yahoo [35], as well as our own research [8] confirm that SRMs are common at large scale experimentation. During this study, and through quantitative analysis of experiments conducted within the last year we identified that approximately 6% of experiments at Microsoft exhibit an SRM. We illustrate this in Figure 2 where we show the variation of the SRM ratio among five products that run experiments at a large scale (ordered in descending order based on the number of experiments per product). Figure 2 reveals that this is an important problem to address as it happens frequently – for example, a product running ten thousand experiments in a year can expect to see at least one SRM per day.

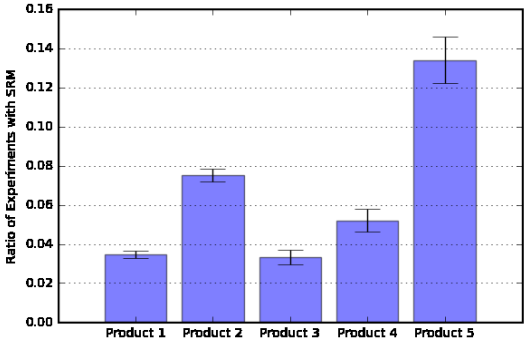

Figure 2. Ratio of OCEs with an SRM at Microsoft.

4.2 SRMs Indicate Invalid Analysis

Each of the experienced analysts confirmed that (1) they are regularly exposed to SRMs and have to identify their root-causes, and that (2) SRMs are a critical data quality issue that needs to be investigated and resolved before a conclusive decision on the result of an experiment can be reached. In particular, SRMs cause a selection bias that invalidates any causal inference that could be drawn from the experiment. If there is a selection bias in treatment or control sample, then the observed metric movements may be due to the selection bias and cannot be attributed to the treatment effect. Below are two quotes from our interviews that affirm these observations:

> *"I can recall hundreds of SRMs. We consider it* ***one of the most severe*** *data quality issues we can detect."*

> *"I am working on resolving an SRM just now. The SRM* ***is critical.*** *The analysis is* ***completely untrustworthy****."*

4.3 SRMs are Challenging to Root-Cause

Analysts at our case companies report a large variance for the time that it takes to investigate an SRM, and many components of the experiment in which SRMs could happen. The time to diagnose an SRM can span from minutes to months, and SRMs can happen due to issues with experiment design, the randomization process, variant deployment, data logging, data collection, data cooking, as well as due to data analysis conditions.

> *"Most of the time I spend a day or two to resolve an SRM. I would look at the* ***deep dive,*** *look at the* ***correct filter condition*** *etc. If it is a browser related SRM* ***it may take me a week,*** *if it is not browser related* ***it can take me months****."*

The quotes above illustrate the high complexity and effort needed to investigate an SRM throughout the experiment. We expand on the individual components in section 5 in greater detail.

4.4 SRM Indicators make Diagnosis Effective

While a simple statistical test can be used to indicate whether experiments has an SRM or not, it cannot provide enough clues to experimenters on what the root cause may be. To make the

analysis efficient, analysts at our case companies employ detectors that help them in the SRM investigation process. These detectors span from specialized tools for investigating individual segments of users or time slices of an experiment, as well as specialized data quality metrics that can be used as leading indicators of a root cause of an SRM. We illustrate this finding with example quotes next:

"My most common tool is looking at the chart of assignments to treatment and control ***over time****, at a fairly low granularity, e.g. 1 hour. The differences either in the beginning or in the middle of the experiment indicate the time of SRM."*

"We have created a bunch of data quality metrics that can guide us. For example, we have a metric called a proportion of users with 0-page views. ***If we change the rate in which people click on the home page*** *you will see that the* ***proportion of users with 0-page views goes up****."*

4.5 Some SRMs Can Be Prevented

While avoiding SRMs completely is not feasible in practice, certain types of SRMs can be prevented. Simple SRMs caused by e.g. redirects or filtered conditions can be prevented by educating experimenters on designing their experiments more correctly. For example, experiments can redirect both control and treatment variations of a web page when a redirection in one of them is needed, they can also select a broader triggering condition, prevent humans from interfering with experiment variants, and so on. We illustrate this with quotes below and provide a more comprehensive list of preventive actions in section 5.

"Many SRMs were caused by ***a human intervening into the experiment and unknowingly breaking it****. This could be prevented by introducing controls that show warnings or block users from making changes to running experiments."*

4.6 SRMs Can Have a Positive Cause

While most SRMs occur due to issues in one or more of the experiment stages, some SRM can be an indicator of positive treatment behavior. For example, when the load speed of the treatment variant is improved, the likelihood of the load event being logged increases. Similarly, if the treatment variant increases engagement of the users with the product (for example, users are forced to click on two items as opposed to only one in control) – the likelihood of losing both events during telemetry collection is smaller compared to only losing the single event.

"If the ***treatment is getting us to recover additional users*** *in the logs* ***it almost always*** *means that it made some* ***performance*** *improvements to the website. So, these are wins!"*

The findings and example data in this section are only a small subset of the knowledge that we constructed about SRMs during this study. In the next section, we provide an overview of the common SRM types we inferred from this data.

5. FIVE COMMON TYPES OF SRMs

In this section, we provide an overview of the common SRM types and a taxonomy revealing the different categories of SRMs based on the stage of experiment where they appear. For each of the SRM types, we first provide an example experiment that resulted in a puzzling outcome. Next, we explain why the experiment exhibited an SRM and provide a general understanding of the scenario. Finally, we discuss how to detect and prevent the SRM scenario where applicable. We summarize all the common causes with a taxonomy on Figure 5 in Section 5.6.

5.1 Experiment Assignment SRMs

Experiment. For every A/B test, it is a best practice to compute results for the pre-period before the test started and confirm that a random split of users into two groups does not cause a large difference in key metrics. If it does, then the same metric in the A/B test period would carry over some of that bias.

During an experiment on MSN.com webservice, the team observed an SRM in one such A/A test. How can an A/A test have an SRM?

Outcome. Further investigation revealed a *bug* in the experiment assignment service. The experiment assignment service used a *hash function* to randomize users into one thousand buckets, each bucket representing 0.1 percent of users. For each experiment, the experiment owners define how much user traffic they want to expose each variant in the experiment to, which translates into a specific number of buckets to assign to control and treatment. In this case, there was a *bug in the experiment assignment service* where the control variant was assigned one less bucket than the required number of buckets. A 50/50 test would incorrectly be setup as a 49.9/50 split, which is not necessarily an obvious issue.

Generalization. Experiment assignment service is one of the fundamental components of an experimentation platform [21]. There are three main requirements for this component. First, end users must be equally likely to see each variant of an experiment (assuming a 50-50 split). Second, repeat assignments of a single end user must be consistent i.e. the user should be assigned to the same variant on each successive visit to the product. Third, when multiple experiments are run, there must be no correlation between experiments. An end user's assignment to a variant in one experiment must not have effect on the probability of being assigned to a variant in any other experiment.

Detection. Violation of any requirement can cause an SRM. Problems in the experiment assignment service would usually cause multiple experiments to have an SRM. In our example above, the first condition was violated. *Unstable user ids* can cause violation of the second requirement and may lead to an SRM in certain situations. The third requirement can be violated in more subtle ways. For example, it is important that an experiment assignment is not correlated with the experiments that are running alongside the experiment, or with the experiments that ran before. If a treatment variant of an experiment is particularly good (or bad) then it may affect the re-visit rate of users during the time the experiment is running and may have *a carryover effect* even after the experiment stops. If this experiment's assignment is correlated with another experiment's assignment, then it may lead to an SRM.

Prevention. In many cases experiment variant assignment is done by *hashing the user ID*. With different seeds, hashing libraries like MD5 lead to hashing functions that are independent of each other. If the experiment assignment service is picking a seed at random from a list of seeds, it is important to make sure that there is a very large number of seeds to ensure no two experiment assignments are correlated. For 365 different seeds, you only need around 23 experiments to have a 50% chance of at least one pair of experiments sharing a seed. This is also referred to as birthday paradox [27]. Further it is important to change the list of seeds used over time, or else correlation in user behavior can build up for users who are assigned to the same bucket by a hashing function. In some cases, it is essential to make sure that

experiment assignment of an experiment is not independent of another experiment but rather exclusive of another experiment. When variants in two different experiments interact with each other, randomizing these experiments independently may cause an SRM. For instance, if the application crashes when a user is in treatment for two different experiments together and these two experiments are randomized independently of each other, then it may lead to an SRM due to telemetry loss. To avoid an SRM in this case, no user should be exposed to both experiments at the same time i.e. experiment assignments are exclusive of each other. This can be achieved by using a separate population for each experiment or by running experiments in different time frames.

5.2 Experiment Execution SRMs

Experiment. The team at Skype recently ran an OCE aimed at improving audio quality of calls. Varying network conditions are the largest driver of poor audio quality experience in voice over IP (VoIP) calling applications such as Skype. Dynamically adjusting buffering parameters can help improve audio quality. However, there are multiple buffering parameters, and their optimal setting may depend on a variety of network characteristics. The treatment variant used a context-based machine learning (ML) model to set the buffering behavior of the received audio stream. The control consisted of a fixed parameter used for all call conditions. The ML algorithm used an online system to learn the buffering parameters across contexts by trading off playout delay and distortion.

Outcome. The team anticipated an improvement in audio metrics by using a context-aware parameter value compared to a fixed setting. Instead, they found that the experiment resulted in a significant increase in audio distortion and playback delay. The randomization unit of this experiment was a session (call). The experiment analysis system collected 30% less sessions compared to the control group, triggering an SRM alert.

Explanation. Upon investigation it was revealed that the root cause of the Skype experiment was an asynchronous refresh of the experimentation configuration in the middle of a session. The findings were based purely on systems metadata. While the updated configuration was not honored in the middle of the session (by design), the in-memory variable tracking the variant ID was getting updated due to a bug. As a result, the experimentation log was recording and reporting the incorrect variant ID at the end of the session. Around 30% of the logs reported as if a treatment never started as the treatment. After detecting this issue, the team could evaluate the experiment for a subset of the population – the sessions that did not encounter a configuration refresh in the middle of a session.

Generalization. The root cause above can be classified into a category of SRM scenarios that we label – Experiment Execution SRMs. This scenario consists of three main categories: *(1) SRM's due to Variant delivery behavior* (e.g. starting variants at different times), *(2) SRM's due to Variant execution behavior* (e.g. due to delayed filter execution which targets populations based on complex features that are only known during the experiment execution), and *(3) SRM(s) due to Telemetry generation behavior.* For the latter, there are several root-causes. First, *experiments not redirecting all variants* of a web page may have an SRM as some redirects may fail. Second, if *new telemetry is added to the product*, the likelihood of receiving at least one event back from one-time users increases, which typically results in observing more users in that variant. Third, if the *treatment degrades performance* of a product, an SRM can happen as users have more time to exit the product before the logs are generated. In contrast, improved product performance can give telemetry generation component more time to generate and send logs, frequently resulting in observing more users in the faster variant.

Another common SRM point in this stage is is with *first run* occurrences. SRMs can happen in first run experiments whenever one variant increases the engagement of users with the product (e.g. a variant has a bug that only engaged users discover), if the treatment changes the order in which its components are loaded (e.g. users exit the product before the telemetry is generated, or *if a variant has a bug and crashes* (see OneNote experiment in [12]). Logs from a faster or crashing variant will be delayed or, worse, may never be generated and sent for processing. A similar observation can be seen with experiments that make *users more engaged* with the product (e.g. they make more clicks). In such experiments, the likelihood of transmitting at least one log is higher, reducing telemetry loss rate and potentially resulting in an SRM with more one-time users.

Furthermore, to process and report on the results of an experiment the system needs to collect telemetry from all places where the experiment was executed. Since the experiment may be executed on several physical devices, data may be *lost in transport* to a central processing pipeline. This is especially likely to occur in cases where telemetry is generated on third party devices, such as cell phones or in browsers. For example, threads may be closed or paused by the client operating system to save battery power or network may be unreliable or even entirely disconnected during experiment execution. There are many ways in which telemetry may go missing between a client and the experiment system. *When a treatment affects the rate at which data goes missing* an SRM may be the result. For example, if the treatment is a more promising compressing algorithm that uses less bandwidth to achieve the same result, the probability of telemetry loss in the collection may decrease as a side effect of the treatment.

Detection. Finding the root cause of an SRM in the Experiment Execution stage is far from trivial. One way to diagnose the exact root cause is to have data quality metrics measuring telemetry loss, product performance metrics measuring the speed and reliability of actions, and engagement metrics measuring users' activity. On Figure 3 below, we provide an example alert from Booking.com illustrating one of the data quality metrics (counting the number of warnings) and a warning revealing that the proportion of users between the variants is not as expected.

			request_warning_count - Number of warnings triggered			
717707 (1h9m)	Exp. visitors	Goal server requests	Data validity	Sum	Exp. average	Goal average
base (50%)	31 167 (49.66%)	22 (13.41%)	[BASE]	24 warnings	770.0452×10^{-6} warnings	1.0909 warnings
variant 1 (50%)	31 594 (50.34%)	142 (86.59%)	More data points in this variant. This is bad, and the values are no longer comparable.	143 warnings	4.5262×10^{-3} warnings	1.007 warnings

Figure 3. Alert detecting telemetry loss at Booking [23].

For more details on diagnosing SRMs in this stage see [13]. Note however, that sometimes-advanced approaches may be needed to identify the exact root-cause. For example, in the Skype audio experiment, the team built a classifier to detect when the algorithm's output was invalidated (class=1) by the telemetry system and when it was validated (class=0). The algorithm was a random forest model and looked at the most discriminative features. By using this classifier, they found that the top features were all related to the session duration. The longer the session, the more susceptible it is to configuration refresh. This finding gave the necessary hint to trace it back to the experimentation configuration refresh. In general, the team finds this approach of

using a feature difference between the two classes to be quite helpful in finding clues regarding the source of the SRM when the cause is not immediately obvious.

Prevention. Systemic SRMs (e.g. like the one with Skype experiment above) can be prevented by tracking the telemetry reliability and evaluating any bias between control and treatment variants [13]. The telemetry loss metrics will not help prevent experimental design SRMs – these SRMs should be prevented during the experimental design phase and it requires a careful understanding and evaluation of whether the variant would change the underlying population between control and treatment. Our overall recommendation to prevent some of these SRMs is to introduce first telemetry signals, which should fire before any other code in the product executes. This helps register user's presence in a more reliable way. First signal, however, should be used in conjunction with other data quality metrics to avoid making any biased conclusion in situations with data loss.

5.3 Experiment Log Processing SRMs

Experiment. To explain this scenario, we return to the experiment increasing the number of rotating cards on the carousel from 12 to 16 described in the introduction of this paper.

As explained earlier, an increase in the number of times that users click and engage with the carousel was expected, but the controlled experiment revealed a significant decrease in engagement with the cards instead. Crucially, the treatment group which was exposed to the new 16 card carousel had less users in the analysis compared to what was configured and expected during experiment setup, triggering an SRM warning. How did this happen?

Outcome. A deep-dive analysis by experienced analysts revealed that during the data processing phase, the most engaged users in the treatment group were being classified as computer bots and removed from the experiment analysis by an algorithm that scans for bot activity. Some users in the treatment engaged with the carousel cards so much that they crossed the engagement threshold used in the bot detection algorithm. After accounting for the bot classification results, the results were flipped and the correct decision to ship the feature was made.

Generalization. We classify SRMs caused by problems during log processing, such as this example, into a third category dubbed Experiment Log Processing SRMs. In this stage, most SRMs happen due to issues caused by the way the data is processed, or "cooked": it is *transformed from raw telemetry into summary statistics* on various levels (session level, user level). For that, the data is *aggregated, joined, filtered or summarized*. Automation of that process might fail, for instance leaving blanks in the data that goes missing. *Removal of bots* during the cooking phase based on post-treatment observations can lead to SRM when treatment affects the bot detection system such as in the example described above. Other examples of SRM caused during this phase include incorrect joins resulting in silently dropping data points from the analysis. Some of these issues could be considered simply bugs or bad experiment design, but often their impact is magnified when they result in SRM depending on the nature of the treatment.

Many of the problems that fall into Log Processing can be considered as "missing data" problems. The impact of the missing data on the analysis depends whether the data is missing because of a transient issue happening after data collection, and thus by repeating the cooking process maybe the data can be recovered. If data is missing because it was not collected at all, it is often impossible to resolve the issue without repeating the entire experiment. Conversely, if data is missing because of bad joins, filtering, or due to a delay in telemetry transmission, then it may be possible to repeat the cooking process after the problem has been fixed. However, this will depend on system architecture, as it may not be possible to repeat the cooking process at all in stream processing systems where the necessary raw data is not stored long enough.

Detection. One method for detecting issues related to collection and cooking is described by Kaufman et al. [15]. If there are two or more distinct data collection and cooking pipelines it is very unlikely that telemetry collection or telemetry cooking issues are identically replicated in all pipelines. Simply comparing the summary statistics resulting from different pipelines will help identify when SRM is likely caused. Further inspection of different parts of the pipelines can help hunt down the exact root cause. Another good practice for detecting SRMs in Log Processing is to implement monitoring of the size and types of log records that get excluded from the analysis (e.g. by monitoring 'users' filtered out as bots).

Prevention. Data collection issues can never be avoided entirely. As they are especially prevalent when dealing with third party systems one potential solution is to ensure that users are not exposed to treatment before they are triggered into the experiment on a local system. For example, by ensuring that mobile devices require explicit confirmation from the experiment platform that a user has been enrolled into the experiment (see First Signal in 5.2 prevention), or by automatically triggering users into the experiment when experiment configuration is sent to their device, even when it is not yet known whether they will be exposed at all.

Many problems that occur during cooking could be described as post-treatment selection effects. Filtering out bots based on data generated after treatment exposure can be problematic when treatment affects signals used by the bot detection system. To prevent these kinds of issues, the cooking process should ensure that no post-treatment data is used for filtering purpose. For example, at Booking.com, data processing pipelines determine user attributes, such as their bot detection status, at first exposure to the experiment and disallow changes after this point (for that experiment). This way, post-treatment selection issues are entirely avoided.

5.4 Experiment Analysis SRMs

Experiment. Microsoft Teams recently ran an experiment testing whether skipping a First-Run Experience (FRE) page affected user activity. An FRE is the page that users see when they open the product for the first time. Triggered analysis was applied for the users who saw FRE based on the events fired when the control users saw the FRE page, or the treatment users were expected to see it.

Outcome. There were more users in treatment variation in the triggered scorecard, while no SRM appeared in the non-triggered (aka standard) analysis that includes all users of the product.

Explanation. To save network communication, the client sends out the events in batches. The old design took longer to load, which lead users in control to quit earlier. As a result, the probability that the data logs that indicate the visibility of the old

design being lost was higher. The main finding was that the trigger condition can't cover all the eligible users for the analysis.

Generalization. The SRM issue above falls into the category of Experiment Analysis SRMs. In this stage, *telemetry filtering* is the main root cause, which frequently happens because of incorrect configuration of analysis or incorrect set up of counterfactual logging. Incorrect triggering or filtering condition is another common case. Another example is a situation that a feature is displayed to users in treatment only, but the counterfactual logging for users in control is missing, making it hard to drill-down.

Detection. If no SRMs happen with non-triggering or non-filtering analysis, it indicates that the root cause lies with the triggering or filtering analysis process. For example, to solve the problem in the Microsoft Teams experiment presented earlier, the team needed to configure the analysis based on the events with a relaxed condition that happened a few steps before the change with the new design. They determined users' eligibility in the triggered analysis to see the FRE page based on events logged immediately after the users log in. Zhao et al. [35] introduced a method of calculating the cumulative sample size from the very beginning of experiment to diagnose whether starting time point lead to SRM. Chen et al. [4] mentioned that computing SRM ratios for first time users and return users also helped identify the effect of retention changes. Missing data from a variant in the filtering analysis indicates the condition is invalid or counterfactual logging is missing.

Prevention. To prevent Telemetry Filtering SRMs, we recommend starting the analysis from the beginning of the experiment. If it is hard to balance sample size with the triggering or filtering condition, try relaxing the condition to include a wider range of users. For example, generate analysis for users who have seen a view page instead of users who have taken actions on the page, when the later condition can cause bias. Ideally, implement a framework to deal with common counterfactual logging scenarios, such as apply the same mechanism for counterfactual logging when display new view pages or new elements on a page.

5.5 Experiment Interference SRMs

Experiment. In one of the experiments at Microsoft Store Homepage and Microsoft Homepage, the team evaluated the impact of the page redesign (see Figure 4 for example screenshots of two variants). The experiment had an SRM despite being designed correctly, the variants logging the correct data, and all the other root causes were dismissed. For some reason, however, there were more users in the control variation than expected.

Outcome. After a deep-dive analysis, the team learned about a human interference with the experiment variant. Some users were forced into one of the variants by a search engine campaign which had a misconfigured URL pointing directly to the variant (as opposed to the product). The solution to resolve the particular SRM was simple – to get an insight into the results, the team needed to segment the affected users out of the analysis and provide experiment results for the users that were not forcefully assigned. In most cases, however, rerunning the experiment will be necessary.

Generalization: The root cause above classifies into a category of SRM scenarios that we label – Experiment Interference SRMs. This scenario consists of two categories: (1) *SRMs due to Variant Interference* - these happen due to an involvement of an experimenter or the actual end user in the experimentation process. The example above was an example of an experimenter interaction in placing users from a search engine campaign into one of the experiment variants. Many products have hidden way to *"force" a specific variant experience* using config strings or URL parameters for debugging purposes which, if not used correctly, can cause an SRM. Another example experienced by our case companies is a situation where an experimenter would pause one of the variants for some time during the experiment to perform a change, or a situation where only some of the variants are ramped-up.

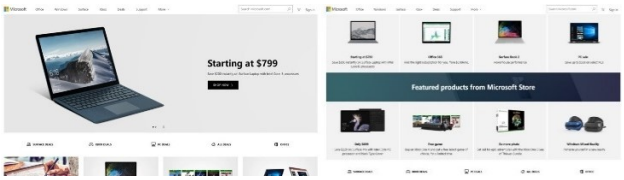

Figure 4. Microsoft Store and Homepage Experiment.

The second category are (2) *SRMs due to telemetry interference* – These are cases where *users of the product actively interfere* with the product variation in a way that results in an SRM. For example, in one experiment at one of our case companies, users actively manipulated the product telemetry by submitting an injection attack through one of the telemetry fields. This caused a severe SRM in the variant that the user was conducting the attack.

Detection. To detect SRM's due to experimenter's involvement, we recommend establishing monitoring of activities with the platform and alerting whenever a variant is changed during experiment execution. To detect forceful assignment to experiment variants, monitor the assignment of users over time and alert the experimenter on large and significant differences.

Prevention. This type of SRMs can effectively be prevented by notifying the experimenters about the consequences of their actions. On Figure 5, we provide an example alert from an experimentation platform used at Outreach.io that notifies the experimenter about the consequences of stopping a running variant during an experiment.

To prevent Interference SRMs we recommend preventing experimenters from using direct assignments to experiment variants outside of certain network (e.g. only allow team members to be directly assigned a variant), alerts in the experimentation platform that notify experimenters about the consequences of their actions, and to monitor the telemetry that is being collected for unusual values (blanks, injection strings, etc.).

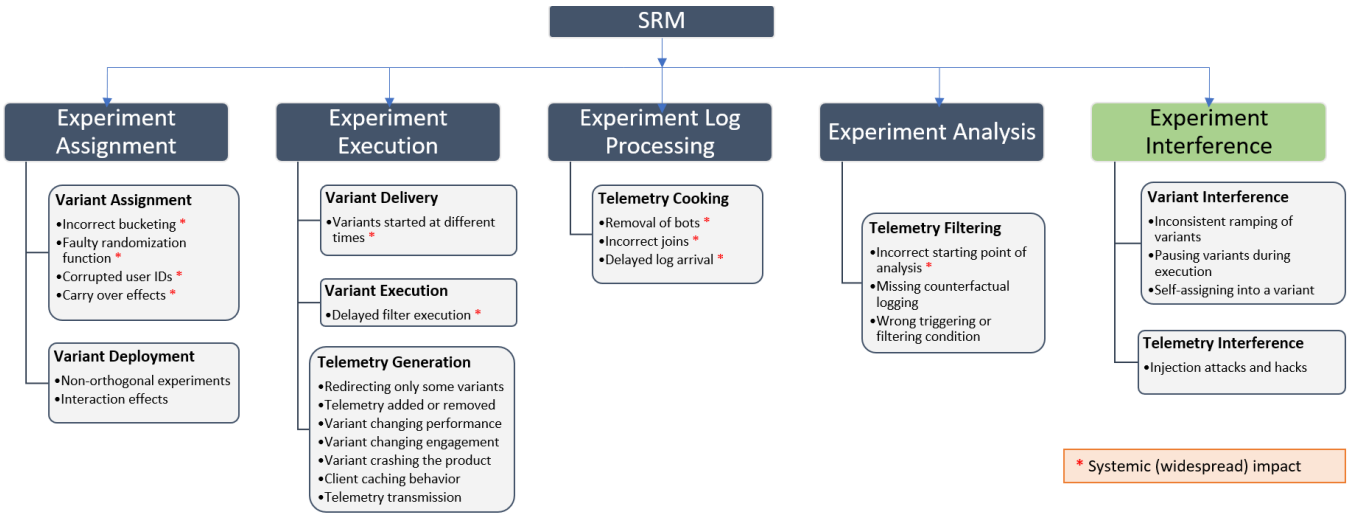

Figure 5. Taxonomy of common SRM types and root-causes.

5.6 SRM Taxonomy

We summarize the common SRM causes that we discussed so far on Figure 6. In total, we recognize 25 distinct causes of SRMs and categorize them based on the stage of the experiment in which they appear. We colored the root-causes that likely indicate a systematic widespread impact (e.g. impacting multiple experiments) with *. In the next section, we discuss the rules of thumb for classifying the SRM that you experience and determining the root cause.

6. Rules of Thumb for SRM Investigations

In the previous section, we provided the SRM taxonomy and example experiments that exhibited an SRM. With this contribution, practitioners can increase their awareness of common SRMs and learn about preventive measures employed by our case companies. In this section, we briefly discuss ten rules of thumb for a quick diagnosis and categorization of most SRMs. This will help practitioners narrow down the root cause and categorize the SRM.

1. **Examine scorecards**: If SRM occurs in a subsample of users (e.g. in a triggered/filtered scorecard) and there is no SRM in the standard scorecard with all users in the experiment, it is likely that the trigger/filter condition is wrong in experiment analysis. Start by relaxing the filter to capture a wider audience and examine at what point the SRM problem occurs.
2. **Examine user segments:** If the SRM occurs in only one user segment, it is very likely that the cause of the SRM is localized to that segment. For instance, if the treatment is relying on some advanced browser capabilities like 256-bit encryption [25], the segment with older version of the browser might have an SRM.
3. **Examine time segments:** If the evidence for SRM is strongest on day 1 of the experiment, and you do not observe the SRM after a longer time duration, the SRM is likely due to time related factors like caching during experiment execution, or delayed start of a variant in the experiment.
4. **Analyze performance metrics**: If there is a large degradation in key performance metrics like time to load or crashes in the scorecard that has an SRM, then it is likely that the sign of this difference is real and is causing the SRM. For instance if the treatment increases page load time, then you may not get telemetry from some users who get the worst page load time, but you will still see regression when comparing treatment with control for those users for whom you do have telemetry.
5. **Analyze engagement metrics**: If average engagement per user is higher in treatment compared to control, then it is likely that the root cause of the SRM is affecting less engaged users more, and vice-versa. For instance, in the case of Skype SRM example the root cause was due to a system bug impacting sessions with longer (more engaged) duration.
6. **Count frequency of SRMs**: If many disparate experiments have an SRM, then it is likely that the root cause of the SRM is a systemic issue due to one or more factors from the taxonomy with a systematic widespread impact (marked *).
7. **Examine AA experiment:** If an A/A experiment has an SRM then it is likely that one of the widespread factors is the root cause of the SRM, or the experiment is not actually an AA – for instance if you add extra telemetry in one of the variants then it is not an A/A as that variant might recover more users.
8. **Examine severity:** If you observe a very large or very small sample ratio, then the root cause is likely to be affecting most users in the control or treatment, respectively. For example, if there are no users in one variation, then it is likely a telemetry issue where control variant or a trigger condition in control is not getting logged properly.
9. **Examine downstream**: If your pipeline allows introspection of data at different collection and aggregation stages (e.g. in steps before the final scorecard), then comparing results at different stages may provide clues as to where the SRM originates.
10. **Examine across pipelines:** If your experimentation system has two data pipelines, then compare the results of both pipelines. See [15] for more details. Also, examine debugging logs containing records that could not be merged in the pipelines. Differences in these point to log processing related SRMs.

7. CONCLUSION

The biggest driver of incorrect conclusions when comparing two datasets is bias. SRMs are a strong indicator of such bias, are common in large scale experimentation, and are difficult to diagnose and resolve. They consume analyst time, attribute to flawed conclusions, reduce confidence in experimentation, and impede the progress of product development. In this paper, based on our experience of running OCEs in four different software companies in over 25 products used by hundreds of millions of users, we derived a taxonomy for the types of SRMs and general rules of thumb for diagnosing their root cause. We hope that the lessons learned and practical tips for diagnosing each of them will raise awareness of this important topic. This should prevent some of the SRMs from happening as companies mature their experimentation practices as well as speed-up their resolution.

ACKNOWLEDGMENT

The authors of this paper would like to thank the interviewees and all the case companies that participated in this study. We would also like to acknowledge everyone that reviewed this paper and provided guidance and feedback.

REFERENCES

[1] Auer, F. and Felderer, M. 2018. Current State of Continuous Experimentation: A Systematic Mapping Study. *Proceedings of the 2018 44rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (Prague, Czechia., 2018).

[2] Bakshy, E., Eckles, D. and Bernstein, M.S. 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.

[3] Barik, T., Deline, R., Drucker, S. and Fisher, D. 2016. The Bones of the System: A Case Study of Logging and Telemetry at Microsoft. (2016).

[4] Chen, N., Liu, M. and Xu, Y. 2018. Automatic Detection and Diagnosis of Biased Online Experiments. *arXiv preprint arXiv:1808.00114*. (2018).

[5] Deng, A., Lu, J. and Litz, J. 2017. Trustworthy Analysis of Online A/B Tests. *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining - WSDM '17* (New York, New York, USA, 2017), 641–649.

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

[7] Dmitriev, P., Frasca, B., Gupta, S., Kohavi, R. and Vaz, G. 2016. Pitfalls of long-term online controlled experiments. *2016 IEEE International Conference on Big Data (Big Data)* (Washington, DC, USA, Dec. 2016), 1367–1376.

[8] Dmitriev, P., Gupta, S., Dong Woo, K. and Vaz, G. 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).

[9] Fabijan, A., Dmitriev, P., Holmstrom Olsson, H. and Bosch, J. 2018. Effective Online Controlled Experiment Analysis at Large Scale. *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (Prague, Czechia., Aug. 2018), 64–67.

[10] Fabijan, A., Dmitriev, P., McFarland, C., Vermeer, L., Holmström Olsson, H. and Bosch, J. 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.

[11] Fabijan, A., Dmitriev, P., Olsson, H.H. and Bosch, J. 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).

[12] Fabijan, A., Dmitriev, P., Olsson, H.H. and Bosch, J. 2017. The Benefits of Controlled Experimentation at Scale. *Proceedings of the 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (Vienna, Austria, Aug. 2017), 18–26.

[13] Gupchup, J., Hosseinkashi, Y., Dmitriev, P., Schneider, D., Cutler, R., Jefremov, A. and Ellis, M. 2018. Trustworthy Experimentation Under Telemetry Loss. *to appear in: Proceedings of the 27th ACM International on Conference on Information and Knowledge Management - CIKM '18* (Lingotto, Turin, 2018).

[14] Gupta, S., Ulanova, L., Bhardwaj, S., Dmitriev, P., Raff, P. and Fabijan, A. 2018. The Anatomy of a Large-Scale Experimentation Platform. *2018 IEEE International Conference on Software Architecture (ICSA)* (Seattle, USA, Apr. 2018), 1–109.

[15] Kaufman, R.L., Pitchforth, J. and Vermeer, L. 2017. Democratizing online controlled experiments at Booking. com. *arXiv preprint arXiv:1710.08217*. (2017), 1–7.

[16] Kevic, K., Murphy, B., Williams, L. and Beckmann, J. 2017. Characterizing Experimentation in Continuous Deployment: A Case Study on Bing. *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)* (May 2017), 123–132.

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

[18] Kohavi, R., Deng, A., Frasca, B., Walker, T., Xu, Y. and Pohlmann, N. 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.

[19] Kohavi, R., Deng, A., Longbotham, R. and Xu, Y. 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, New York, USA, 2014), 1857–1866.

[20] Kohavi, R. and Longbotham, R. 2017. Online Controlled Experiments and A/B Testing. *Encyclopedia of Machine Learning and Data Mining*. C. Sammut and G.I. Webb, eds. Springer US. 922–929.

[21] Kohavi, R., Longbotham, R., Sommerfield, D. and Henne, R.M. 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.

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

[23] Leaky Abstractions: 2018. *https://booking.ai/leaky-abstractions-in-online-experimentation-platforms-ae4cf05013f9*.

[24] Lindgren, E. and Münch, J. 2015. Software development as an experiment system: A qualitative survey on the state of the practice. *Lecture Notes in Business Information Processing* (Cham, May 2015), 117–128.

[25] List of browsers that support 128-bit and 256-bit encryption: .

[26] Mayring, P. 2002. Qualitative content analysis - research instrument or mode of interpretation. *The Role of the Researcher in Qualitative Psychology*. 139–148.

[27] Mckinney, E.H. 1966. Generalized Birthday Problem. *The American Mathematical Monthly*. 73, 4 (Apr. 1966), 385. DOI:https://doi.org/10.2307/2315408.

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

[29] MIT Code: 2016. *http://bit.ly/Code2016Kohavi*.

[30] Pearson, K. 1900. X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*. 50, 302 (Jul. 1900), 157–175. DOI:https://doi.org/10.1080/14786440009463897.

[31] Runeson, P. and Höst, M. 2008. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*. 14, 2 (2008), 131–164. DOI:https://doi.org/10.1007/s10664-008-9102-8.

[32] SRM Interview Guide: 2019. *https://www.dropbox.com/s/h0291u1fcqg0eze/SRM Interview Guide.pdf?dl=0*.

[33] Tang, D., Agarwal, A., Brien, D.O., Meyer, M., O'Brien, D. and Meyer, M. 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.

[34] Xu, Y., Chen, N., Fernandez, A., Sinno, O. and Bhasin, A. 2015. From Infrastructure to Culture. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15* (New York, New York, USA, 2015), 2227–2236.

[35] Zhao, Z., Chen, M., Matheson, D. and Stone, M. 2016. Online Experimentation Diagnosis and Troubleshooting Beyond AA Validation. *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)* (Oct. 2016), 498–507.