CHALMERS
UNIVERSITY OF TECHNOLOGY

Engineering for a science-centric experimentation platform

Citation for the original published paper (version of record):
Diamantopoulos, N., Wong, J., Issa Mattos, D. et al (2020). Engineering for a science-centric experimentation platform. Proceedings - International Conference on Software Engineering: 191-200. http://dx.doi.org/10.1145/3377813.3381349

(article starts on next page)

Engineering for a Science-Centric Experimentation Platform

Nikos Diamantopoulos
ndiamantopoulos@netflix.com
Netflix, Inc.
Los Gatos, California, USA

Jeffrey Wong
jeffreyw@netflix.com
Netflix, Inc.
Los Gatos, California, USA

David Issa Mattos
davidis@chalmers.se
Chalmers University of Technology
Gothenburg, Sweden

Ilias Gerostathopoulos
gerostat@in.tum.de
Vrije Universiteit Amsterdam,
Netherlands and Technical University
of Munich, Germany

Matthew Wardrop
mawardrop@netflix.com
Netflix, Inc.
Los Gatos, California, USA

Tobias Mao
tmao@netflix.com
Netflix, Inc.
Los Gatos, California, USA

Colin McFarland
cmcfarland@netflix.com
Netflix, Inc.
Los Gatos, California, USA

ABSTRACT

Netflix is an internet entertainment service that routinely employs experimentation to guide strategy around product innovations. As Netflix grew, it had the opportunity to explore increasingly specialized improvements to its service, which generated demand for deeper analyses supported by richer metrics and powered by more diverse statistical methodologies. To facilitate this, and more fully harness the skill sets of both engineering and data science, Netflix engineers created a science-centric experimentation platform that leverages the expertise of scientists from a wide range of backgrounds working on data science tasks by allowing them to make direct code contributions in the languages used by them (Python and R). Moreover, the same code that runs in production is able to be run locally, making it straightforward to explore and graduate both metrics and causal inference methodologies directly into production services.

In this paper, we provide two main contributions. Firstly, we report on the architecture of this platform, with a special emphasis on its novel aspects: how it supports science-centric end-to-end workflows without compromising engineering requirements. Secondly, we describe its approach to causal inference, which leverages the potential outcomes conceptual framework to provide a unified abstraction layer for arbitrary statistical models and methodologies.

CCS CONCEPTS

• **Software and its engineering → Software design engineering**;

KEYWORDS

experimentation, A/B testing, software architecture, causal inference, science-centric

ACM Reference format:
Nikos Diamantopoulos, Jeffrey Wong, David Issa Mattos, Ilias Gerostathopoulos, Matthew Wardrop, Tobias Mao, and Colin McFarland. 2020. Engineering for a Science-Centric Experimentation Platform. In *Proceedings of Software Engineering in Practice, Seoul, Republic of Korea, May 23–29, 2020 (ICSE-SEIP '20),* 10 pages.
https://doi.org/10.1145/3377813.3381349

ICSE-SEIP '20, May 23–29, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7123-0/20/05…$15.00
https://doi.org/10.1145/3377813.3381349

1 INTRODUCTION

Understanding the causal effects of product and business decisions via experimentation is a key enabler for innovation [15, 29, 30, 37], and the gold-standard of experimentation design is the randomized controlled trial, also known as A/B testing [5, 46, 48].

In this paper, we will be presenting key aspects of the experimentation platform built by Netflix, a leading internet entertainment service. The innovations of this experimentation platform are interesting because they have resulted in a "technical symbiosis" of engineers and data scientists, each complementing the skill sets of the other, in order to create a platform that is robust and scalable, while also being readily extensible by data scientists.

Netflix routinely uses online A/B experiments to inform strategy and operation discussions (e.g. [4, 21, 31, 35]), as well as whether certain product changes should be launched. Over time these discussions grew to be increasingly specialized, generating demand for more and richer metrics powered by extensible statistical methodologies that are capable of answering diverse causal effects questions. For example, it was becoming more common for teams to require bespoke metrics to assist in the analysis of specific experiments, such as when changes to Netflix's UI architecture and video player design caused extra hard-to-isolate latency in playback startup [21]; or to require bespoke statistical methodologies, such as when interleaving was used to garner additional statistical power when trying to compare two already highly-optimised personalization algorithms [4].

To support these ever-growing use-cases, Netflix made a strategic bet to make their experimentation science-centric; that is, to place a heavy emphasis on enabling arbitrary data analyses methods for causal inference that are developed in different fields of science. To implement this science-centric vision, Netflix's experimentation platform, `Netflix XP`, was reimagined around three key tenets: trustworthiness, scalability, and inclusivity. Trustworthiness is essential since results that are untrustworthy are not actionable. Scalability is required to accommodate for Netflix's growth. Inclusivity is a key tenet because it allows scientists from diverse backgrounds such as biology, psychology, economics, mathematics, physics, computer science and other disciplines, working on data science tasks, to contribute to the experimentation platform.

The implications of these tenets on `Netflix XP` are wide-ranging, but perhaps chief among them are the resulting choices of language and computing paradigm. Python was chosen as the primary language of the platform; with some components in C++ and R as needed to support performance and/or statistical models. This was a natural choice because it is familiar to many data scientists, and has a comprehensive collection of standard libraries supporting both engineering and data science use-cases. The platform also adopted a non-distributed architecture in order to reduce the barrier of entry into the platform for new statistical methodologies. Since non-distributed architectures are not as trivially scaled, the techniques employed by the platform in order to ensure scalability, i.e. compression and numerical performance optimizations, are a significant contribution of this work.

The re-imagined `Netflix XP` has also had implications for its stakeholders. Firstly, data science productivity has increased. It is now straightforward for data scientists to reproduce and extend the standard analyses performed by the experimentation platform because they can run the production code in a local environment. The code also permits ad hoc extensions, allowing scientists to leverage their background and domain knowledge to easily deliver customized scorecards [14]; for example, by including explorations of heterogeneous or temporal effects. Secondly, data science workflows have been enriched with a more extensive toolkit. Since the platform was re-imagined, new statistical methodologies, such as quantile bootstrapping and regression, have been contributed to the platform, which can then be used in combination with arbitrary metrics of the data scientists' choice. Thirdly, engineers have been freed up to focus on the platform itself. Since data scientists are now responsible for contributing and maintaining their own metrics and methodologies, engineers are now able to focus on aspects of the platform in which they specialize, leading to greater scalability and trustworthiness. The effect of these implications has compounded in rapid innovation cycles around ongoing strategy discussions, which has changed the face of experimentation at Netflix.

In this paper, we provide two main contributions. Firstly, we report on the architecture of this platform, with a special emphasis on its novel aspects: how it supports science-centric end-to-end workflows without compromising the engineering requirements laid out in subsequent sections. Secondly, we describe its approach to causal inference, which leverages the potential outcomes framework to provide a unified abstraction layer for arbitrary statistical models and methodologies.

The rest of this paper is organized as follows: Section 2 presents background information in online experiments and related work. Section 3 presents the research method and validity considerations. Section 4 presents the architectural requirements, the libraries and improvements made to `Netflix XP` to support science-centric experimentation and the impact of these changes at Netflix. Section 5 discusses the causal inference framework used by `Netflix XP` that allows scientists to express their causal models in a unified way. Section 6 concludes the paper and discusses future research directions.

2 BACKGROUND AND RELATED WORK

2.1 Online Experiments

Online experiments have been discussed in research for over 10 years [3]. The most common type of online experiment is the randomized controlled trial (RCT). RCT consists of randomly assigning users to different experience (control and treatments) of the product, while their behavior is gauged via logging a number of events. Based on this telemetry, several metrics are computed during and upon completion of an experiment. Statistical tests, such as the t-test, Mann-Whitney test, or CUPED [9] are used to identify statistically significant changes in the metrics and generate scorecards [14]. These scorecards help product managers, engineers, and data scientists to make informed decisions and identify a causal relationship between the product change and the observed effect. RCT in web systems is extensively discussed by Kohavi et al. [29]. The paper presents an in-depth guide on how to run controlled experiments on web systems, discussing types of experimental designs, statistical analysis, ramp-up, the effect of robots and some architecture considerations, such as assignment methods and randomization algorithms.

Although most research in online experiments has focused on RCT, companies have been using other types of experimental designs to infer causal relations. For instance, Xu and Chen [47] describe the usage of quasi A/B tests to evaluate the mobile app of LinkedIn. The paper details the characteristics of the mobile infrastructure that contribute to the need for designing and running different experiment designs than RCT.

2.2 Experimentation Processes and Platforms

To support and democratize experimentation across multiple departments, products and use cases, Kaufman et al. [27] have identified the need for an experimentation platform to be generic and extensible enough to allow the design, implementation, and analysis of experiments with minimal ad hoc work. They describe, in the context of Booking.com, the usage of an extensible metric framework to enable experiment owners to create new metrics. However, they do not describe the extensibility aspect in the context of different experimental designs and analyses as we do.

Twitter discusses its experimentation platform and how it is capable of measuring and analyzing a large number of flexible metrics [11]. The platform supports three types of metrics: built-in metrics that are tracked for all experiments, event-based metrics, and metrics that are owned and generated by engineers. One of the challenges is to scale the system with this flexibility. Scalability was achieved through several performance optimizations in their

infrastructure including profiling and monitoring the capabilities in Hadoop and making processing jobs more efficient.

The trustworthiness aspect of online experiments has been an active area of research [10, 16, 25, 28, 49]. Experiments that rely on violated assumptions or are susceptible to implementation or other design errors can lead to untrustworthy results that can compromise the conclusions and the value of the experiment. Kohavi et al. [28] discuss lessons learned from online controlled experiments that can influence the experiment result, such as carryover effects, experiment duration, and statistical power. Fabijan et al. [16] provide essential checklists to prevent companies from overlooking critical trustworthiness aspects of online experiments. In our work, we do not specifically focus on trustworthiness aspects of online experiments, but on how to make the experimentation process science-centric.

More similar to our work, the different software architecture parts and design decisions of an experimentation platform are presented in Gupta et al. [23]. The paper describes the core components of the Microsoft ExP Platform, focusing on trustworthiness and scalability. In summary, their platform can be divided into four main components: experimentation portal, experiment execution service, log processing service, and analysis service. In the platform, experiment owners can easily create, deploy, and analyze experiments reliably and at scale. The platform also supports deep-dive post-experiment analysis for understanding metric changes in specific segments. However, such analysis requires a deep understanding of the structure of the data, the computation of the metrics, and the way experiment information is stored in the data. In our work, we specifically focus on the analysis components of `Netflix XP` and describe how they have been re-designed to allow science-centric experimentation.

2.3 Enhancing Productivity of Data Scientists

Finally, re-designing `Netflix XP` to afford data scientists the ability to work with their familiar languages and tools is akin to other efforts to enhance the flexibility and productivity of data scientists. A prominent example is Tempe [18], an integrated, collaborative environment for large-scale data analytics, which allows for both offline and real-time analytics using the same scripts written in a scripting variant of C#. Tempe relies on a temporal streaming engine, Trill [7], to progressively compute and report analysis results. It also relies on the concept of *live programming* in which statements are re-evaluated upon edit to keep the results of a script up-to-date with the script text. Our work also aims at providing a homogeneous environment for both ad hoc and production analysis flows. To achieve interactivity, `Netflix XP` does not rely on a temporal streaming engine, but on a combination of pre-computing tables and on-demand slicing. Also, contrary to Tempe, interactive analysis of experiments in `Netflix XP` is performed in Jupyter Notebooks and follows the classic read-eval-print loop [8].

3 RESEARCH METHOD

Netflix is an entertainment media service provider and content producer with over 150 million subscribers. Within the scope of the online streaming platform, Netflix runs hundreds of experiments yearly. `Netflix XP` has been running and supporting experiments at Netflix for over 9 years.

Over the last 3 years, an increased need for flexibility in the design and analysis of experiments, as well as the need to optimize the usability of the platform for its data scientists has led Netflix to redesign its experimentation infrastructure as science-centric. The redesign followed an engineering process that resembles the steps proposed in design science research [24, 36]. Design science seeks to investigate and develop new and innovative artifact solutions that emerge from the interactions of the operating environment, organization technology, and involved stakeholders. The artifact resulted from such process and described in this research is the (`Netflix XP`). The development of the artifact was subjected to continuous evaluation based on feedback from the data scientists and software developers using the platform to deploy online experiments.

This paper reports on the main aspects of the platform produced that reinforced the key tenets of the platform—trustworthiness, scalability and inclusivity. To identify these main aspects, data from multiple sources was collected between the years of 2017 and 2019. The primary source of data consists of documentation from three regularly scheduled meetings: Experimentation Engineering with Experimentation Science leaders, Experimentation Science strategy meetings, and Experimentation Engineering with Experimentation Science verticals. These meetings provided the main aspects in which the platform was evaluated. Additionally, we collected data from the company-wide summit on forward thinking plans for experimentation, one-on-one interviews with data scientists, engineers, and product managers, internal experiments, observational data from the usage of the platform as well as software documentation and product roadmap documents.

The collected data was analyzed in three steps. In the first step, we gathered all the major design changes of `Netflix XP`. In the second step, we coded these changes into common groups [6], such as requirements, architecture changes, software libraries, performance improvements, statistical methods, and causal inference modeling. Similar codes were merged and grouped under the two main themes discussed in this paper: the software architecture and the causal inference framework. We classified within each theme the changes that produced, and are expected to produce, high impact for the platform. The relative impact of the changes was assessed based on direct feedback observed in the collected data, e.g. recurring feedback mentioned in meetings and in one-on-one interviews. These feedback were also supported by internal experiments and observational data. We then staged the changes in a way that made the foundations of the `Netflix XP` strong.

Validity considerations. *External validity*: This study is based on a single company and the results and decisions taken that led to changes in the architecture, software libraries, performance improvements and causal modeling are dependent on the specific context of the development activity. However, the presented results can provide guidance to other organizations seeking to evolve a science-centric experimentation culture, since not all of the presented results and discussions are tied directly to `Netflix XP` or to the streaming service industry. *Instantiation validity* refers to the validity of the proposed artifacts in the specific context of design science [32]. One of the main threats is the large artifact instantiation

space, where the artifact can be instantiated in a large number of design options. While there are other ways that the experimentation platform could be instantiated, the artifact context and the inclusivity tenet led the platform development to be based on existing tools, implementations and theoretical frameworks [33].

4 SOFTWARE ARCHITECTURE

Due to the ever increasing number of simultaneous experiments, experimentation platforms are often expected to derive conclusions without much human intervention. While automation brings a huge boost in velocity, Netflix's view is that it should not stand in the way of custom analyses that can leverage domain expertise in order to improve the understanding and context of the effects created by an experiment. Online experiments can easily become very complex and challenging to analyze [10]. In such cases, the custom designs and analyses made by the involved data scientists are of great importance.

The large number of data scientists running custom analyses required Netflix to redesign its experimentation platform. From Netflix's experience, when the stakeholders who need some changes are not empowered to make them, the results are sub-optimal. When facing engineering barriers to integrate with production systems, data scientists might end up creating isolated solutions that may not be integrated with the production system. This can lead to multiple fragmented systems with different degrees of documentation and levels of support.

This section describes the architectural components in `Netflix XP` to support science-centric experimentation. These components give data scientists full autonomy to run their analyses end-to-end, and empower them with the necessary software tools to do deep-dive analyses. We first describe the requirements and architecture of the platform and then illustrate how it enables data scientists to perform deep-dive analyses and contribute new analysis flows.

4.1 Requirements for Experiment Analysis

Regarding the analysis of online experiments, `Netflix XP` has the following requirements:

- **Scalable**. Each experiment at `Netflix XP` may collect and analyse data from a large portion of its 150M subscribers. Since Netflix is a very fast growing business this is just the starting point for the scalability requirement.
- **Performant**. Experiment results must be calculated within seconds or minutes to allow for exploratory analyses with different user segments and metrics.
- **Cost efficient**. The use of computational and storage resources for experiment analysis must be minimized to avoid unnecessary costs.
- **Trustworthy**. `Netflix XP` must offer reproducible results with accurate calculations on all statistics.
- **Usable**. Data analysts must be able to effortlessly specify standard and specialized analysis flows, view the results in an intuitive graphical interface, and perform custom deep-dive analyses.
- **Extensible**. Data scientists from different backgrounds must be able to easily extend `Netflix XP` by contributing new experimental designs and analyses.

The last two points are directly related to science-centric experimentation. They imply that data scientists can easily setup a development environment so they can reproduce, debug, and extend analyses that happen in production. This development environment should in particular:

- support interfacing with existing scientific libraries, such as Pandas, Statsmodels, and R;
- support local and interactive computation, for example through Jupyter Notebooks.

4.2 Experiment Analysis Flow

An experiment analysis at `Netflix XP` consists of three distinct phases: data collection, statistical analysis, and visualization of the results [34]. These phases, along with the related components of the platform, are depicted in Figure 1. As a first step, experiment log data are extracted, enhanced by user metadata, and stored as a table in S3. The resulting table is subsequently filtered, aggregated, and compressed based on the specified analysis configuration. Those first two tasks are achieved through the `Metrics Repo` component which is responsible for generating the appropriate SQL expressions that will be run on top of Spark and Presto. Second, different statistical methods are run on the compressed data to calculate the specified metrics and statistics of interest for the experiment. This step is performed by the `Causal Models` component. Third, different graphs are plotted for visual analysis of the results—a task of the `XP Viz` component. All of the above are orchestrated via the `XP Platform API` which is responsible for delivering the calculated metrics and produced graphs to `Netflix XP`'s frontend, `ABlaze`. Alternatively, an identical analysis can also run within a Jupyter Notebook; in this case, the analyst can further customize the analysis and view the results directly in the Notebook environment.

To enable science-centric experimentation, all three main components of `Netflix XP`'s architecture can be extended by data scientists providing new metrics, statistical methods, and visualizations. These three components are described in detail next.

4.3 Metrics Repo

`Metrics Repo` is an in-house Python framework where users define metrics as well as programmatically generated SQL queries for the data collection step. One of the main benefits of this is the centralization of metric definitions in a unified way. Previously, many teams at Netflix had their own pipelines to calculate success metrics which caused fragmentation and discrepancies in calculations.

For each metric, the framework allows contributors to define certain metadata (e.g. the statistical model to use and how to be displayed). In order to compare a metric across two user groups, aggregate data of users in each group need to be collected and compared. For example, Figure 2 shows the specification of a "number of streamers" metric: for each user, the number of streaming sessions with duration more than one hour are collected. For comparison between user groups, a default set of descriptive statistics, as well as proportion tests are used.

Many related systems in the industry generate the required data for analyses through a rigorous Extract, Transform, Load (ETL) pipeline which is responsible to annotate the available business data with the experiment data. A key design decision of `Metrics`

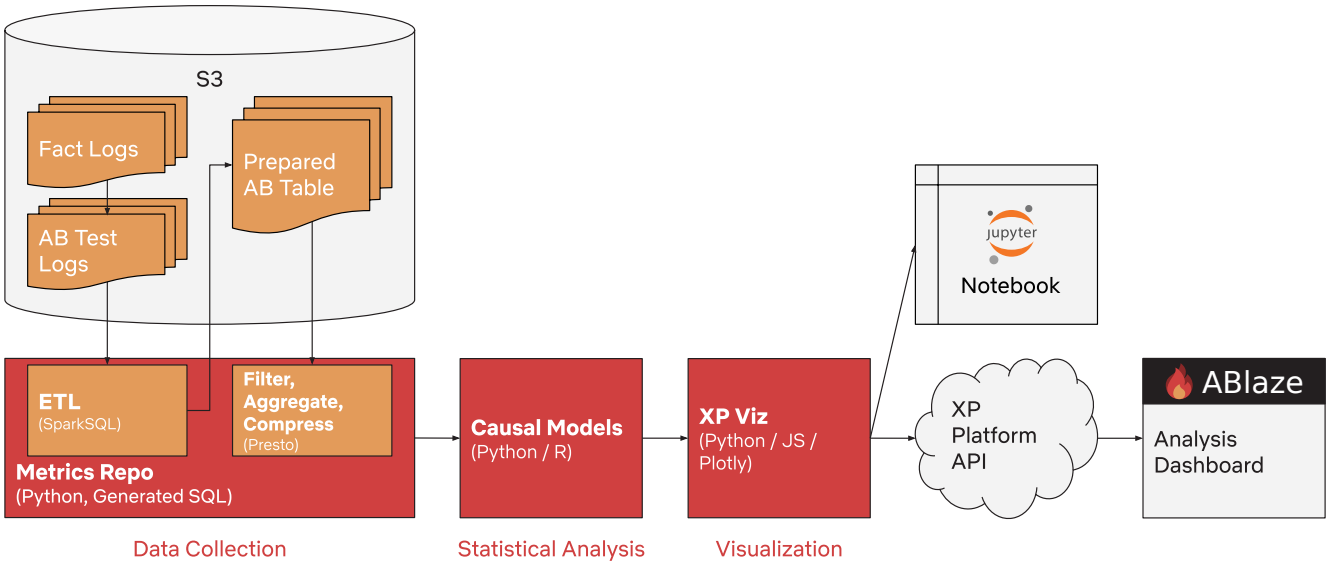

Figure 1: Experiment analysis flow at `Netflix XP`.

```
class NumStreamers(Metric):

  def _expression(self):
    return Max(If_(self.query.streaming_hours > 0, 1, 0))

  def _statistics(self):
    return [DescriptiveStats(), ProportionStats()]
```

Figure 2: Example Metric definition.

Repo is that it moves the last mile of metric computation away from data engineering-owned ETL pipelines into dynamically generated SQL that runs on Spark. This allows data scientists to add metrics and join arbitrary tables in a faster and much more flexible way since they do not have to conform to a strict predefined schema. The generated SQL is run only on demand and on average takes a few minutes to execute. This ad-hoc data collection removes the need for migrations and expensive backfills when making changes to metrics avoiding the costly and slow ETL alternative. Adding a new metric is as easy as adding a new field or joining a different table in SQL. The SQL is generated programmatically in Python which leads to a maintainable and self-documented code base.

4.3.1 Pre-compute vs Live-compute. When analyzing an experiment, data scientists need to see the metrics through different slicing of their data. Slicing is typically done based on different dimensions, e.g., user's country (only US users) or device type (only iOS users). To support this in the past, statistics would be computed for each dimension value over all dimensions (pre-compute). Such computation leads to an explosion of possible comparisons: e.g. statistics for users in each country are compared separately, statistics for users on different device types are again compared separately. The problem becomes exponential when slicing is applied via conjunction of dimension values (e.g. US users on iOS) or dis-junction (e.g. users from US or Canada) due to the number of possible combinations.

To cope with the above problem, the platform adopts the following hybrid solution. When a new analysis is requested, statistics for only a number of commonly used slices are pre-computed. If more slices are needed, the respective statistics are computed on demand (live-compute). Live computation is not instant, however, with an average latency of less than a minute, it is easy to queue all the slices and view the results as they become available within seconds. To achieve the above, the data collection is split into two steps: the first one retrieves raw data without filtering and aggregations, whereas the second retrieves the final set of filtered and aggregated data. The first part is usually much more costly to compute (multiple minutes) due to big table joins, so it is calculated in Spark and stored as a table in S3. The resulting table can subsequently be sliced with the requested filter on demand. For quick slicing over large amounts of data, Presto [40], a distributed SQL engine, has been used due to its fast and interactive nature in computing filter and aggregate queries compared to alternatives such as Spark or Hive.

Lastly, it is worth noting that, given that comparing all the pre-computed slices is statistically controversial due to multiple hypothesis testing [17], `Netflix XP` offers segment discovery through `Causal Models`, which enables automatic discovery of important data slices instead of manually comparing them one by one.

4.3.2 Building trustworthiness. `Metrics Repo` comes with two powerful features that increase the confidence in changes. The first is a testing framework which, besides unit testing, allows integration testing where metric calculations run end-to-end on real sample data leveraging Spark. This enforces that every change goes through a continuous integration system, ensuring that none of the well established reports are affected. Contributors are given the appropriate tools and are urged to follow internal best practices

when submitting changes. The second feature is the option to run a meta-analysis on historical tests with the proposed changes. This enables contributors to change a metric definition and view how this would have affected hundreds of completed tests, allowing them to confidently decide if they should move forward. Those two features have proven valuable in providing a safety net and a solid base for changes.

4.4 Causal Models

`Causal Models` is a Python library that houses implementations of causal effects models and serves as the statistical engine for `Netflix XP`. Causal effects models are a restricted class of statistical models that measure causation instead of correlation, a distinction that is crucial in the context of experimentation. `Causal Models` receives data from `Metrics Repo`, then reports summary statistics such as the mean, count, and quantiles under a model, as well as treatment effect statistics such as the average treatment effect, its variance, its confidence interval and its p-value. Like `Metrics Repo`, the library is designed for inclusion in that it allows scientists to contribute causal effects models that integrate into the experiment analysis workflow. It also leverages the same meta-analysis framework as `Metrics Repo` to ensure stability across changes. To support the management of many models, the `Causal Models` library also employs a modeling framework for causal inference, though we defer that discussion until section 5 and focus on `Causal Models` as a mechanism for statistical testing here.

Netflix seeks to utilize a full repertoire of causal effects models from different scientific fields in order to provide rich data for decision making. The two-sample t-test is the most foundational causal effects model in AB testing. It is simple to understand, simple to implement, is easily scaled, and measures causal relationships instead of correlational ones when the data is randomized and controlled. Building on top of that, ordinary least squares (OLS) is a causal effects model that can be used to determine the differences in the averages while filtering noise that the t-test cannot filter. Quantile regression can be used to determine differences in quantiles of the distribution, for example if Netflix is concerned about changes in its most engaged users. Panel models can be used to measure treatment effects through time.

By building modeling tools using the same stack that scientists use, `Netflix XP` was able to overcome many challenges in graduating multiple causal effects models. Often, advances in modeling are developed by scientists with in-depth knowledge of statistics, and their methods are usually inspired by domain knowledge and experience in their field. To support their work in field experiments, their models are developed in programming languages such as R and Python that emphasize local and interactive computing. The process of graduating such causal effects models into a production engineering system can be inefficient. First, context and knowledge must be transferred. Afterwards, the models would frequently be re-implemented in Spark in order to make them performant in a big data environment. Implementations in a distributed computing environment, such as Spark, makes models hard to debug, and introduces a high barrier for scientists to contribute. This challenge often leads scientists to create ad-hoc applications in order to communicate their research and conclusions about an experiment. Instead of re-implementing models, `Causal Models` is built on Python, and engineers an interface that can integrate these models into `Netflix XP` while preserving the important engineering requirements discussed in section 4.1. This created a path from research directly into the experiment analysis workflow. In this case, the tenet of being inclusive to the data science stack improved the science-centric vision, as well as the tenet on trustworthiness. The innovations required to reach this milestone are discussed below.

To make contributions easier for scientists, `Causal Models` offers all the necessary support to integrate with existing statistics libraries in Python and R, the most common data science languages at Netflix. Having a multilingual framework makes `Netflix XP` inclusive to scientists from different backgrounds. `rpy2` [20] has enabled the use of R inside a python framework by embedding an R process, but sharing data across them can consume large amounts of memory. In order to minimize RAM usage, the platform employs Apache Arrow [1], an in-memory and cross language data format that offers zero-copy inter-process communication. Additionally, `Causal Models` provides: (1) parallelization over multiple metrics during the calculation of statistics and (2) caching to simplify managing multiple models for multiple metrics.

Integrating with non-distributed Python and R libraries enables single-machine computation that is easy to debug and extend, however this emphasis and deviation from distributed computing can reduce the scalability of the experimentation platform. Therefore, the `Netflix XP` engineering team developed optimizations to scale modeling, so that the stack can still serve production and also offer local computation. This was addressed in two ways: data compression, and high performance numerical computing.

Data compression is an engineering achievement that allows better inclusion of the data science stack, and improves the tenet on scalability. Many causal effects models, such as OLS, compute the difference between the means of two distributions; these means are estimated using averages from a dataset. Some distributions can be losslessly summarized using sufficient statistics [19]. For example, the Normal distribution can be summarized using conditional means and variances. When sufficient statistics are available, causal effects models do not need to be trained on the raw dataset, it can be trained on a much smaller dataset containing the sufficient statistics and features for the model. Compression rates as high as 100x were regularly observed, allowing data that would previously require hundreds of gigabytes of memory to fit in a single machine for local and interactive modeling. Compression is a a core part of the `Netflix XP's` analysis workflows, and is applied to all data. For other causal effects models that cannot be summarized using sufficient statistics, a lossy compression is used with sensible defaults that do not materially impact the precision of the treatment effect, as validated by the platform's meta-analysis framework.

Optimizing numeric computations in `Causal Models` is another way to be inclusive, performant, and reduce the need for distributed computing. At Netflix, there is a focus on high performance numerical computing applied to causal effects. This led to the development of reuseable causal effects primitives to support `Causal Models` through highly optimized and generic functions that are common in causal effects analysis. Scientists can use these optimized primitives to compose and contribute their own analyses of experiments, with fewer concerns for performance. For example, linear models

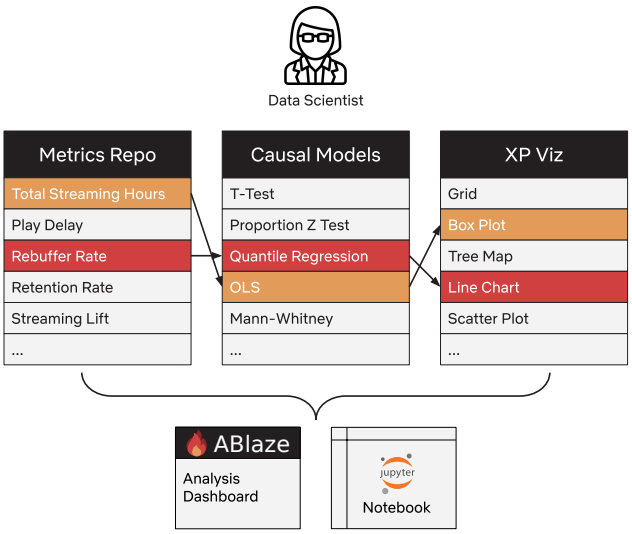

Figure 3: Examples of possible analysis flows in `Netflix XP`.

are a widely used family of causal effects models in `Netflix XP`. They have simple assumptions, are easy to interpret, and are highly extensible: they can be used to estimate average treatment effects, detect segments where treatment effects are different, and measure treatment effects through time. All of these variations can be made faster by using a highly optimized implementation of OLS, which is included in `Causal Models`. Previous work from Netflix XP in [44] demonstrates five significant optimizations to standard implementations of OLS that ultimately can compute hundreds of treatment effects over many millions of users in seconds on a single machine.

Many of the causal effects primitives in `Causal Models` are developed in C++, in order to have low-level control over computation. Although scientists normally use interactive programming languages, many of their primitives are optimized in C or C++, such as the Python library NumPy [41]. C++ enables developers to minimize memory allocations, optimize for cache hits, vectorize functions, and manage references to data without creating deep copies, important aspects of high performance numerical computing. Linear algebra functions that support many causal effects models are invoked through the C++ library, `Eigen` [22]. Using C++, `Netflix XP` engineering can write optimized functions once, and deliver wrappers for python and R through `pybind11` [26] and `Rcpp` [12], maintaining the platform's commitment to inclusivity by supporting a multilingual environment.

By creating an inclusive and scalable development experience for causal effects models, `Netflix XP` has expanded support from two-sample t-tests and Mann Whitney rank tests to many more methods, and has gained confidence that it can include more models that were not originally designed for distributed computing.

4.5 XP Viz

`XP Viz` is the final component of the science-centric experimentation analysis flow in `Netflix XP`. It is a library that provides a lightweight and configurable translation layer from the statistical output of `Causal Models` into interactive visualizations. By implementing it as an independent pluggable component, the platform separates the view layer from the computation layer and allows reuse of standardized visualizations in other contexts. The plotting aspects of the visualization layer is based on Plotly's rich library of graphs components, allowing different teams to reuse, choose, and customize the visualizations of their metrics.

A key benefit introduced by `XP Viz` is that it provides first-class support for Jupyter Notebooks. Data scientists at Netflix regularly use Notebooks for their day-to-day development so supporting their familiar tooling allows them to iterate faster. The integration of `XP Viz` library with Notebooks allows data scientists to not only compute their metrics in a Notebook when exploring, but also visualize them in the exact same way as they would do in the production UI, `ABlaze`. This seamless flow between the server rendered production views to local Notebook rendered views gives data scientists the full power to explore and innovate using the visualizations of their choice.

4.6 Execution of experiment analysis flows

An analysis flow consists of metric definitions from the `Metrics Repo`, corresponding statistical tests from `Causal Models`, and visualizations from `XP Viz`. For instance, a possible flow, depicted in Figure 3, may include the calculation of OLS on total streaming hours and visualize the results using box plots. All of the above steps are orchestrated by `XP Platform API`, a REST API responsible for kicking off computations, keeping state, and storing results.

One of the requirements of the `XP Platform API` is to always function in an interactive manner which means it should remain performant and with consistent latency as computational load increases. Such a requirement becomes more important when multiple users are interacting with the system or when a single user requests multiple slices of an analysis. To achieve this, the heavy computational workload is offloaded to workers instead of using the server processes. This avoids competition for server resources as well as offers a sandboxed environment to run any potentially unsafe code. A common solution in such architectures is to have a list of dedicated machines that are responsible to run the jobs. Instead, Netflix chose to run the jobs on its implementation of OpenFaaS, a serverless computing platform. This solution provides a lot of important features such as autoscaling in a cost effective way, efficient management of the the job queue, managed deploys as well as easy health metric and log collection. Leveraging OpenFaaS provides access to a cluster of machines that guarantees the interactive requirements are met as the load increases.

To unify the execution of workflows with the code in exploratory Jupyter Notebooks, a Notebook-based execution flow is enabled. Essentially, each execution is constructed as a parameterized Notebook that gets evaluated by the different workers. This Notebook can then be extracted and re-run by data scientists to fully reproduce the analysis, which allows them to debug, explore, and extend the analysis as desired. The described Notebook integration creates a natural cycle between the production executions and the ad-hoc Notebook explorations; a production execution can be exported to a Notebook while a Notebook execution can be promoted to production.

4.7 Performing deep-dive analysis

The architecture and framework described above make it possible for data scientists to easily transition from viewing the results in ABlaze to conducting a deeper dive in Notebooks. To illustrate this flow, it is worth revising the NumStreamers metric example (Figure 2) to show how it can be used for further extensions and explorations. After computing an analysis that includes the number of streamers metric, data scientists can view the results in ABlaze and click a button to generate a Jupyter Notebook that replicates the exact same calculations and visualizations [34]. From there, data scientists have multiple potential flows.

Firstly, there is the option of viewing and exploring the raw data or a reasonably sized sample. The data is stored as a Pandas dataframe which offers many easy ways for introspection. This flow is particularly useful in cases where a data scientist wants to get a better sense of the actual values and their distributions in any of the segments they are interested in. On top of that, it is easy to join the data with other tables that were not part of the initially calculated table in order to enrich it with additional information. Such exploratory flows can prove of tremendous importance in analyzing tests as they provide better understanding of the data and increased insight.

Secondly, data scientists can alter the metric definitions and view updated calculations. For instance, a data scientist can redefine the expression for number of streamers to be the people with at least 2 hours of viewing and re-run the statistical analysis.

Thirdly, deep dives can be used to explore the results of different statistical tests other than the predefined ones. This can be achieved by simply adding t-test or OLS in the list of statistics of the metric (Figure 2). Lastly, a data scientist can choose to visualize the results in any of the supported visualizations just by selecting any of the supported plots.

4.8 Contributing new analysis flows

The ecosystem of Metrics Repo, Causal Models, XP Viz, and Jupyter Notebooks not only enables deep dive analysis, it also empowers scientists to use their new learnings to contribute new analyses to Netflix XP. This furthers the "technical symbiosis" where engineers and scientists create a powerful and unified platform together.

Within a Jupyter Notebook, scientists get access to all of the source code from Metrics Repo, Causal Models and XP Viz. This allows them to edit any file they want from within the Notebook, rapidly prototype extensions, and see the impact of the changes. Such a flow could be used to explore improvements to statistics, such as reducing variance on the causal effects. All that is needed is to subclass the causal model base class and conform to the generalized causal models API. In a similar fashion, new visualizations can also be prototyped from within a Notebook. When improvements are discovered during deep dive analysis, they can be iterated on locally and interactively in the notebook, and contributed back into Metrics Repo, Causal Models or XP Viz, which were all designed to be inclusive to scientists of different backgrounds. That improvement becomes available to all experiments in the Netflix XP, which can again be reproduced in a Jupyter Notebook for another scientist to iterate on. This cycle between deep dive, improve, and contribute has led to a culture of rapid iteration and the ability to release many new metrics and analysis flows. Based on the collected observational data from the usage of the platform internally, within less than a year of the introduction of the new architecture, more than 50 scientists have directly contributed more than 300 metrics, as well as models such as OLS, quantile bootstrapping, heterogeneous effects, quantile regression, and time series.

5 FRAMEWORK FOR MEASURING CAUSAL EFFECTS

The science-centric vision has greatly influenced the design of Causal Models. It offers a software framework that is not only performant, as mentioned in section 4.4, but also aligns the implementation of a causal effects model with the science of potential outcomes [38, 39], making contributions from scientists natural. Potential outcomes is a generic framing of causal effects computation that is mirrored in Causal Models' programmatic interface. In this way Netflix XP is able to accommodate many causal effects models without having to worry about the domain specific implementation details of the model.

To demonstrate how potential outcomes can be used to unify the computation of three different types of causal effects that Netflix is interested in, consider the following five statistical variables:

(1) y: The metric that needs to be measured.
(2) X: A binary variable indicating whether a user received treatment.
(3) W: Other features that are useful for modeling the variation in y.
(4) t: A variable for indexing time.
(5) θ: Hyperparameters for a model.

The potential outcomes framework is the thought exercise: what would y be if we apply treatment, and what would y be if we do not apply treatment? In a randomized and controlled experiment where all variables are constant except the treatment, any difference in y must be due to noise, or to the treatment. Furthermore, by using this framing, a variety of treatment effects that are important for a business can be computed from an arbitrary model. The average treatment effect (ATE) on y due to receiving the treatment, X, can be generically formulated as

$$ATE = E[y|X = 1, W] - E[y|X = 0, W].$$

This treatment effect is the expected global difference between the treatment experience and the control experience. Likewise, the causal effect on y due to X for the subpopulation where $W = w^*$ is the conditional average treatment effect (CATE), and can be formulated as

$$CATE(w^*) = E[y|X = 1, W = w^*] - E[y|X = 0, W = w^*].$$

This treatment effect shows opportunities to personalize for different segments. In many cases, the treatment effect needs to be traced through time, which is the dynamic treatment effect

$$DTE(t^*) = E[y|X = 1, W, t = t^*] - E[y|X = 0, W, t = t^*].$$

This treatment effect can show if a causal effect is diminishing, or if it can persist for a long time. All causal effects models in Causal Models can subscribe to this modeling framework.

Many challenging aspects of managing causal effects models are resolved through this software abstraction based on potential outcomes. For example, models can differ in their input, requirements, and assumptions. A two-sample t-test accepts strictly two metrics, one for the control experience and one for the treatment, and requires that the treatment assignment was randomized. Ordinary least squares (OLS) accepts an arbitrary amount of metrics, the treatment assignment, and a set of covariates for the model. It requires that the treatment assignment was conditionally randomized, and that the covariates are exogeneous and full rank [45]. Finally, it assumes that the noise in the metric is normally distributed. Both of these models assume that the observations about users are statistically independent. This assumption prevents them from being applied to time series data, where following the treatment effect through time is important; a variation of these models would have to acknowledge the autocorrelation in the data. All of these models—the t-test, OLS, and time series variations of them—have different formulas for how to determine if an effect is significant, or just noise. Although these individual models vary, they ultimately only need to return output measuring the expected difference in the potential outcomes.

In addition to creating a path to contribute causal effects models and consolidating three types of treatment effects, `Causal Models` is able to implement the boilerplate and reduce the amount of code a scientist needs to contribute. All three variations of treatment effects are differences in potential outcomes, where features of the model are controlled to be specific values. They also use the same procedure: (1) train a model, (2) create a copy of the input dataset where treatment is applied to all users, (3) create another copy where treatment is not applied to any user, (4) predict the potential outcomes from each of these data copies, (5) take the average of the predictions, (6) then difference the averages. Finally, a model must implement another method for computing the variance on the treatment effect, so that it can test if the effect is significant or noise. This procedure is a burden to implement for every causal effects model, but it can be reduced through a simple software interface. Each causal effects model that subscribes to the framework only needs two unique methods, then `Causal Models` completes the work that is common to how all causal effects models compute treatment effects. The interface for an individual model requires methods to:

(1) **Train** a model on a dataset containing y and X, and optionally W, and t.
(2) **Predict** the expected value of y for the potential outcomes $X = 1$, and $X = 0$.

`Causal Models` as a unifying software framework across multiple causal effects models does the work to prepare the data, invoke the train and predict methods, then aggregate and difference the output. Optionally, a model can also implement methods for ATE, CATE, or DTE directly, for example if there is a specialized computational strategy for them, as in [44]. Finally, the bootstrap in [13] allows `Causal Models` to compute the variance for an arbitrary causal effects model.

Developing this software framework honors the scientific study of causal effects, and is another form of engineering that can allow `Netflix XP` to better include work from scientists.

6 CONCLUSION

In this paper we have introduced the architecture and innovations of Netflix's experimentation platform, which is routinely used to perform online A/B testing for the purposes of informing business decisions. The architecture's design was strongly influenced by a strategic bet to make the platform science-centric and support arbitrary scientific analyses on the platform, which led to its being non-distributed and written in Python. Our analysis of the platform has resulted in two novel contributions to the literature: how an experimentation platform can be designed around data science contributions without sacrificing trustworthiness and scalability, and how this is in part achieved by framing the experimentation inference problem generically enough to allow for arbitrary methodologies (via the potential outcomes conceptual framework). Other innovations include compression strategies and low-level statistical optimizations that keep the non-distributed platform performant.

Since the release of the platform described in this paper, there has been a significant increase in the engagement and contributions to the experimentation platform from scientists. This includes not only the local installation of the experimentation platform tooling, but also direct contributions of metrics and methodologies that have greatly enriched the platform and the analyses it can perform. This "technical symbiosis" of engineers and scientists has greatly increased the pace of innovation in experimentation at Netflix, and has already resulted in even deeper strategy discussions around richer analyses.

The next frontiers for the Netflix experimentation platform revolve around feature-based analyses, automation, and adaptive experiments. Feature-based analyses will allow for richer explorations of the treatment effect and interactions between multiple features in a single experiment. Automation will allow for tests to be programmatically created and modified in response to events on the platform. Adaptive experiments leverage the former two features in order to allow for automated decision making during the test; for example, this might be used to stop tests early if we have sufficient evidence [42, 43] or use multi-arm bandits to optimally choose per-feature test allocation rates [2, 50]. Working groups of engineers and scientists have already started collaborating on how to best approach these features in a science-centric manner.

We hope that reporting this case study will spark interest in science-centric experimentation platforms, and welcome feedback from companies or individuals interested in working or collaborating on this important topic.

ACKNOWLEDGMENTS

This work was partially supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and by the Software Center. This work was partially sponsored by the German Ministry of Education and Research (grant no 01Is16043A).

We also acknowledge Martin Tingley for supporting the science-centric vision, and Rina Chang, Pablo Lacerda de Miranda, Susie Lu, Sri Sri Perangur, and Michael Ramm for their substantial contributions to the experimentation platform.

REFERENCES

[1] Apache Arrow. 2019. *Apache Arrow. A cross-language development platform for in-memory data.* https://arrow.apache.org/

[2] Susan Athey and Stefan Wager. 2017. Efficient Policy Learning. arXiv:math.ST/1702.02896

[3] Florian Auer and Michael Felderer. 2018. Current state of research on continuous experimentation: A systematic mapping study. In *Proceedings - 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018.* 335–344. https://doi.org/10.1109/SEAA.2018.00062

[4] Juliette Aurisset, Michael Ramm, and Joshua Parks. 2017. *Innovating Faster on Personalization Algorithms at Netflix Using Interleaving.* https://medium.com/netflix-techblog/interleaving-in-online-experiments-at-netflix-a04ee392ec55

[5] George Box, Stuart Hunter, and William Hunter. 2005. *Statistics for Experimenters.* Wiley.

[6] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (jan 2006), 77–101. http://www.tandfonline.com/doi/abs/10.1191/1478088706qp063oa

[7] Badrish Chandramouli, Jonathan Goldstein, Mike Barnett, Robert DeLine, Danyel Fisher, John C. Platt, James F. Terwilliger, and John Wernsing. 2014. Trill: a high-performance incremental query processor for diverse analytics. *Proceedings of the VLDB Endowment* 8, 4 (Dec. 2014), 401–412.

[8] Robert DeLine and Danyel Fisher. 2015. Supporting exploratory data analysis with live programming. In *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).* IEEE, Atlanta, GA, 111–119.

[9] Alex Deng, Ya Xu, Ron Kohavi, and Toby Walker. 2013. Improving the sensitivity of online controlled experiments by utilizing pre-experiment data. In *Proceedings of the sixth ACM international conference on Web search and data mining.* ACM, 123–132.

[10] Pavel Dmitriev, Somit Gupta, Dong Woo Kim, and Garnet Vaz. 2017. A Dirty Dozen. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*, Vol. Part F1296. ACM Press, New York, New York, USA, 1427–1436.

[11] Dmitriy Ryaboy. [n.d.]. Twitter experimentation: technical overview. https://blog.twitter.com/engineering/en_us/a/2015/twitter-experimentation-technical-overview.html

[12] Dirk Eddelbuettel and Romain François. 2011. Rcpp: Seamless R and C++ Integration. *Journal of Statistical Software* 40, 8 (2011), 1–18. http://www.jstatsoft.org/v40/i08/

[13] Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap.* CRC press.

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

[15] Aleksander Fabijan, Pavel Dmitriev, Helena Holmstrom Olsson, and Jan Bosch. 2017. The Evolution of Continuous Experimentation in Software Product Development: From Data to a Data-Driven Organization at Scale. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE).* IEEE, Buenos Aires, 770–780. http://ieeexplore.ieee.org/document/7985712/

[16] Aleksander Fabijan, Pavel Dmitriev, Helena Holmström Olsson, Jan Bosch, Lukas Vermeer, and Dylan Lewis. 2019. Three Key Checklists and Remedies for Trustworthy Analysis of Online Controlled Experiments at Scale. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP).* 1–10.

[17] Alessio Farcomeni. 2008. A review of modern multiple hypothesis testing, with particular attention to the false discovery proportion. *Statistical methods in medical research* 17, 4 (2008), 347–388.

[18] Danyel Fisher, Badrish Chandramouli, Robert DeLine, Jonathan Goldstein, Andrei Aron, Mike Barnett, John C Platt, James F Terwilliger, and John Wernsing. 2014. Tempe: An Interactive Data Science Environment for Exploration of Temporal and Streaming Data. (2014), 7.

[19] Ronald A Fisher. 1922. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 222, 594-604 (1922), 309–368.

[20] Laurent Gautier. 2019. *rpy2 - R in Python.* https://rpy2.bitbucket.io/

[21] Corey Grunewald and Matt Jaquish. 2018. *Modernizing the Web Playback UI.* https://medium.com/netflix-techblog/modernizing-the-web-playback-ui-1ad2f184a5a0

[22] Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

[23] Somit Gupta, Lucy Ulanova, Sumit Bhardwaj, Pavel Dmitriev, Paul Raff, and Aleksander Fabijan. 2018. The Anatomy of a Large-Scale Experimentation Platform. In *2018 IEEE International Conference on Software Architecture (ICSA).* IEEE, 1–109. https://doi.org/10.1109/ICSA.2018.00009

[24] Hevner, March, Park, and Ram. 2004. Design Science in Information Systems Research. *MIS Quarterly* 28, 1 (2004), 75. https://doi.org/10.2307/25148625

[25] David Issa Mattos, Pavel Dmitriev, Aleksander Fabijan, Jan Bosch, and Helena Holmström Olsson. 2018. An Activity and Metric Model for Online Controlled Experiments. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* Vol. 11271 LNCS. 182–198. http://link.springer.com/10.1007/978-3-030-03673-7{_}14

[26] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. 2019. pybind11 – Seamless operability between C++11 and Python. https://github.com/pybind/pybind11.

[27] Raphael Lopez Kaufman, Jegar Pitchforth, and Lukas Vermeer. 2017. Democratizing online controlled experiments at Booking.com. *arXiv:1710.08217 [cs]* (Oct. 2017). http://arxiv.org/abs/1710.08217 arXiv: 1710.08217.

[28] Ron Kohavi, Alex Deng, Brian Frasca, Roger Longbotham, Toby Walker, and Ya Xu. 2012. Trustworthy online controlled experiments: Five Puzzling Outcomes Explained. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '12.* ACM Press, New York, New York, USA, 786. http://dl.acm.org/citation.cfm?doid=2339530.2339653

[29] Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal M Henne. 2009. Controlled experiments on the web: Survey and practical guide. *Data Mining and Knowledge Discovery* 18, 1 (2009), 140–181.

[30] Ron Kohavi, Diane Tang, and Ya Xu. 2020. *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing.* Cambridge University Press, Cambridge, United Kingdom ; New York, NY.

[31] Gopal Krishnan. 2016. *Selecting the best artwork for videos through A/B testing.* https://medium.com/netflix-techblog/selecting-the-best-artwork-for-videos-through-a-b-testing-f6155c4595f6

[32] Roman Lukyanenko, Joerg Evermann, and Jeffrey Parsons. 2014. Instantiation validity in IS design research. In *International Conference on Design Science Research in Information Systems.* Springer, 321–328.

[33] Roman Lukyanenko, Joerg Evermann, and Jeffrey Parsons. 2015. Guidelines for establishing instantiation validity in IT artifacts: A survey of IS research. In *International Conference on Design Science Research in Information Systems.* Springer, 430–438.

[34] Toby Mao, Sri Sri Perangur, and Colin McFarland. [n.d.]. Reimagining Experimentation Analysis at Netflix. https://medium.com/netflix-techblog/reimagining-experimentation-analysis-at-netflix-71356393af21

[35] Nick Nelson. 2016. *The Power Of A Picture.* https://media.netflix.com/en/company-blog/the-power-of-a-picture

[36] Jay F. Nunamaker, Minder Chen, and Titus D. M. Purdin. 1990. Systems Development in Information Systems Research. *J. Manage. Inf. Syst.* 7, 3 (Oct. 1990), 89âĂŞ106.

[37] H. H. Olsson and J. Bosch. 2014. From Opinions to Data-Driven Software R D: A Multi-case Study on How to Close the 'Open Loop' Problem. In *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications.* 9–16.

[38] Donald B Rubin. 2005. Causal Inference Using Potential Outcomes. *J. Amer. Statist. Assoc.* 100, 469 (2005), 322–331.

[39] Jerzy Splawa-Neyman, Dorota M Dabrowska, and TP Speed. 1990. On the application of probability theory to agricultural experiments. Essay on principles. Section 9. *Statist. Sci.* (1990), 465–472.

[40] Martin Traverso. 2016. *Presto: Interacting with petabytes of data at Facebook.* https://www.facebook.com/notes/facebook-engineering/presto-interacting-with-petabytes-of-data-at-facebook/10151786197628920/

[41] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. 2011. The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering* 13, 2 (2011), 22.

[42] John Whitehead. 1997. *The Design and Analysis of Sequential Clinical Trials, Revised, 2nd Edition.* Wiley.

[43] John Whitehead. 1997. *Group Sequential Methods with Applications to Clinical Trials.* Jennison, Christopher and Turnbull, Bruce W.

[44] Jeffrey Wong, Randall Lewis, and Matthew Wardrop. 2019. Efficient Computation of Linear Model Treatment Effects in an Experimentation Platform. arXiv:stat.CO/1910.01305

[45] Jeffrey Wooldridge. 2010. *Econometric Analysis of Cross Section and Panel Data.* The MIT Press, Chapter 4.2.

[46] Huizhi Xie and Juliette Aurisset. 2016. Improving the Sensitivity of Online Controlled Experiments: Case Studies at Netflix. ACM Press, 645–654.

[47] Ya Xu and Nanyu Chen. 2016. Evaluating Mobile Apps with A/B and Quasi A/B Tests. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, Vol. 13-17-Augu. ACM Press, New York, New York, USA, 313–322. https://doi.org/10.1145/2939672.2939703

[48] Ya Xu, Nanyu Chen, Addrian Fernandez, Omar Sinno, and Anmol Bhasin. 2015. From Infrastructure to Culture: A/B Testing Challenges in Large Scale Social Networks. In *Proc. of KDD'15 (KDD '15).* ACM, 2227–2236.

[49] Zhenyu Zhao, Miao Chen, Don Matheson, and Maria Stone. 2016. Online Experimentation Diagnosis and Troubleshooting Beyond AA Validation. In *Proc. of DSAA'16.* IEEE, 498–507.

[50] Zhengyuan Zhou, Susan Athey, and Stefan Wager. 2018. Offline Multi-Action Policy Learning: Generalization and Optimization. arXiv:stat.ML/1810.04778