Faster Online Experimentation by Eliminating Traditional A/A Validation

Russell Chen, Miao Chen, Mahendrasinh Ramsinh Jadav, Joonsuk Bae, Don Matheson
Oath Inc.
Sunnyvale, CA, USA
{*rrruss, miaoc, mrjadav, bae, donm*}*@oath.com*

***Abstract*—A/B testing is commonly used to evaluate new features of websites and mobile apps. Before the A/B phase of the experiment, one best practice is to do A/A validation, where all experiment groups receive the control experience. A/A validation requires time and effort to carry out. It serves to ensure that there are no pre-existing differences between the control and test groups as well as to verify the data is flowing in as expected. We propose a method of assigning users to buckets such that the likelihood of observing such a pre-existing difference is vanishingly small. The proposed framework allows us to skip A/A validation, run experiments more quickly and develop our products more swiftly.**

***Keywords*-online experimentation, A/A validation**

I. Introduction

Online controlled experiments, commonly referred to as A/B tests, are an indispensable tool for product development at many technology companies [1]–[3]. The subject of how to carry out such experiments rigorously has been extensively elaborated on [4]–[7] and suggestions for improvements are continually being published [8]–[11]. The overarching goal of running experiments is to determine the causal impact of a set of newly introduced features on metrics important to the business.

This causal impact can be formalized with the Rubin causal model [12]. Conceptually, the framework is well established and relatively simple. However, in the online experimentation setting, a lot of work has to go into verifying that the measurement infrastructure and randomization mechanism are working exactly as intended. Small errors can result in large, undesired deviations. Thus, one recommended best practice is to carry out A/A validation [4].

A. A/A validation

In [4], Kohavi et al. recommend that prior to the start of an A/B test, there should be a period where the experiment groups are subject to the same exact control experience. The experience must be identical in every way - UI, sorting algorithms, backend serving stack, etc. This procedure of A/A validation serves two primary purposes:

1) Check that the control and test buckets are free of pre-existing differences. Because each user is assigned to a bucket at random, it is entirely possible that one bucket contains users who are inherently more active on the site. This difference makes it difficult to estimate the true treatment effect during the A/B phase.
2) Check that instrumentation is working correctly and data is being propagated to reporting systems as expected.

The length of time required for A/A validation is not fixed. Ideally, it would vary depending on how much traffic the buckets receive, just as the bucket size and length of the A/B test itself depends on the expected lift and variability of the metrics. Generally, a few days of data is enough for validation. At Oath, we use a rule of thumb of 4 days. Including the time it takes for data to flow into the reporting systems and dashboards, we need a total of 5 days from the time of experiment creation on our internal experimentation platform before the A/B phase can begin. This is a barrier to quick product development.

Furthermore, because a few buckets are expected to fail A/A validation, our internal experimenters typically open more buckets than they think they will need. Significant effort is required for the experiment owner to decide which buckets (if any) should be used for the experiment.

Unlike the A/B phase of the experiment, A/A validation produces no actionable insights about the product or users. It tells us nothing about what we are really interested in – the average treatment effect or lack thereof. Despite this, A/A validation takes time and effort, as described above. These observations led us to explore options to automate the process or, better yet, skip it altogether.

This paper proposes a method of eliminating traditional A/A validation by guaranteeing that the likelihood of observing pre-existing differences between the control and test buckets is vanishingly small. We refer to the method as "ready-to-use A/A buckets". Our approach to eliminate the need for A/A validation is new both to Oath and, we believe, to the internet industry as a whole. We are not aware of any published papers or unpublished manuscripts that propose a similar methodology nor of systems implemented at other companies that achieve the same goal.

B. Outline

The rest of the paper is structured as follows: Section II provides some background, Section III describes the proposed methodology in detail, Section IV presents results of offline simulations, Section V gives an overview of the engineering implementation, Section VI shows an evaluation of the methodology with real, online experiments and Section VII concludes with a summary and discussion of future work.

II. Background

A. Rubin Causal Model

Let Y_i be the variable of interest for user i. This would be a success metric of the experiment such as page views, sessions, clicks or conversions. We define two potential outcomes Y_i^C and Y_i^T as the values of the metric Y if user i had received the control and test experience, respectively. The treatment effect for user i is then

$$\delta_i = Y_i^T - Y_i^C$$

Averaged over the entire population of users $i = 1, \ldots, N$, the average treatment effect we are interested in is

$$\delta = \frac{1}{N}\sum_{i=1}^{N}\delta_i = \frac{1}{N}\sum_{i=1}^{N}Y_i^T - \frac{1}{N}\sum_{i=1}^{N}Y_i^C$$

Of course, it is not possible for any single user i to be subject to both the control and treatment at the same time. Instead, δ is usually estimated by

$$\hat{\delta} = \frac{1}{N_t}\sum_{i=1}^{N_t}Y_i^T - \frac{1}{N_c}\sum_{i=1}^{N_c}Y_i^C$$

where N_t, N_c are the number of users in the treatment and control groups, respectively. $\hat{\delta}$ is unbiased for δ in a randomized controlled experiment because all assumptions of the Rubin causal model are met. In particular, the randomization assures that there are no confounding factors, whether these factors are observed, unobserved or unobservable.

B. Experimentation Infrastructure at Oath

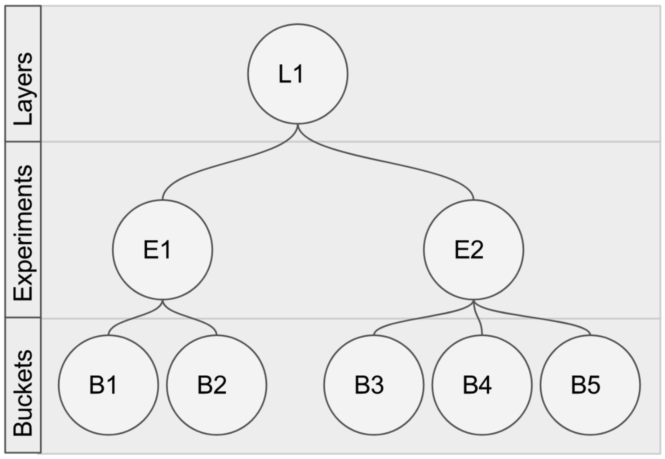


Figure 1. Experimentation Hierarchy

A brief overview of the experimentation system we use internally will be helpful as background for subsequent sections detailing our implementation of ready-to-use A/A buckets.

Most experiments conducted at Oath use the browser cookie as the experimental/randomization unit. This is the type of experiment we will be concerned with here. Cookies are grouped into **buckets** where the exact same experience is served. In order to divert the traffic into the appropriate bucket, each cookie is hashed to an integer in $[0, 999]$. Each bucket is assigned a range of these hash values, where the length of the range corresponds to the requested bucket size. So bucket sizes are specified in increments of 0.1% of the total traffic volume.

Individual buckets are part of an **experiment**. A single experiment has one control bucket and one or more test buckets. All the test buckets in the same experiment are compared to the control bucket in that experiment. Usually each experiment is devoted to testing a single or small set of features. For example, we might have an experiment about the color of a button. The control experience might be “blue button” while the three test buckets might be “purple button”, “green button” and “red button”.

Each experiment (and associated buckets) live on a single **layer**, as shown in Fig. 1. This layer covers the whole range of hash values $0, \ldots, 999$ so the buckets and experiments on the same layer are mutually exclusive, since every cookie is deterministically hashed to the same single hash value. Therefore, the total bucket sizes over all experiments on a given layer may not exceed 100% of the total traffic volume.

By using different random seeds for the hash function, we are able to achieve a multi-layered system, like that described in [13], where each layer captures all the available traffic but traffic splitting is orthogonal across layers. Each user can only fall into at most one bucket/experiment on each layer but possibly multiple buckets/experiments on different layers.

III. Ready-to-use A/A buckets

At a high level, we are going to modify the bucket assignment process slightly by using historical data to create a relatively homogeneous pool of users from which we will randomly draw to create buckets.

First, it is necessary to decide on the metrics important to the business. For our purpose, we chose three user engagement metrics: days visited, a page view (PV)-based and a session-based metric, all computed on a per-cookie basis. These metrics fundamentally quantify product usage both in terms of visitation frequency and level of interaction. They have a long history at Oath with very high data integrity and are important metrics to watch company-wide. For other use cases, these metrics can be substituted as appropriate. In addition, the observed bucket size (count of distinct cookies)

Table I
ILLUSTRATION OF THE READY-TO-USE A/A PROCEDURE

	Rank for each metric			
Hash value	PV	Sessions	Days visited	Count
0	623	426	83	289
🟥 1	135	▨ 33	836	526
2	827	234	931	583
⋮				
🟥 457	389	457	320	▨ 971
⋮				
🟥 998	▨ 19	209	347	836
999	95	883	346	467

should be included for all experiments, giving a total of four metrics.

Note that in the traditional A/A setting, we would carry out 2-sample t-tests to check that there is no statistically significant difference between control/test buckets in the means of the three engagement metrics and a χ^2 goodness-of-fit test to check that there is no evidence the observed difference in bucket sizes is due to anything besides random chance.

Once a small number of metrics are decided on, the following procedure can be carried out daily:

1) For each cookie, compute values of the three engagement metrics for the preceding seven days.
2) Hash all cookies to an integer in $[0, 999]$ and group the cookies by this value.
3) For each hash value, compute the mean values of the three engagement metrics as well as the observed bucket size (count of distinct cookies).
4) Rank the hash values by the means and bucket size computed in (3), for each of the 4 metrics separately.
5) Identify and blacklist all the hash values which rank in the top 50 or bottom 50 for any of the four metrics.
6) The resulting set of hash values is a relatively homogeneous pool with respect to the chosen metrics. Randomly draw an appropriate number of hash values (determined by bucket size requested) from this pool to create buckets.

Table I gives a concrete example of the procedure. All hash values are ranked by each of the four metrics listed. Hash values 1, 457 and 998 (there will be more such hash values not seen here) have at least one rank number ≤ 50 or ≥ 951 (shaded gray) and so will be excluded from consideration (shaded red) when buckets are being created.

Although the hash function distributes users/cookies uniformly across the hash values $0, \ldots, 999$, the means of the engagement metrics by hash value are normally distributed, by a straightforward application of the Central Limit Theorem. If the hash values in the lower tail are combined to form the control bucket while the hash values in the upper tail are combined to form the test bucket, we will see that traditional A/A validation fails. To prevent that from happening, we can just restrict ourselves to looking at the hash values in the central part of the Normal distribution, discarding the tails to form a pool of "mainstream" hash values for consideration in bucket creation. This is what we are doing by blacklisting the top/bottom 50 hash values for each metric. In Table I, hash values $0, 2, \ldots, 999$ belong to the mainstream pool.

The number of hash values blacklisted is not fixed and may vary slightly from day to day. It could be anywhere from 100 (if the top/bottom 50 hash values of each metric are exactly the same) to 400 (if the top/bottom 50 hash values of each of the 4 metrics are completely disjoint). Therefore, the traffic available for experimentation can be between 60% and 90%. The example given in this section removes 50 hash values in each tail. The choice of value for this parameter is discussed in the next section while the stability of the set of mainstream hash values is discussed in Section VI.

IV. SIMULATIONS

The main parameter in the ready-to-use A/A procedure is the amount discarded in the lower/upper tails, where Section III uses 50 for both tails. In order to answer the question of how much to remove and justify our choice of 50, we present simulation results showing how the A/A failure rate varies with the amount removed parameter $tailDiscard$.

Since the distribution of the means is Normal and symmetric, it is intuitively reasonable for the amount removed in each tail to be the same. We have decided to restrict ourselves to that case. The simulation described below first creates the pool of mainstream hash values according to the ready-to-use A/A procedure, then generates "experiments" from this pool.

Each "experiment" consists of one control bucket and a few test buckets, with some restrictions to reflect how buckets are typically created in our internal experimentation platform. The control bucket must be between 1% and 20% while the test bucket is between 1% and the size of the control. We typically do not run buckets smaller than 1% because we will not be able to achieve 80% statistical power in that case. Buckets are rarely larger than 20% in order to constrain exposure of a sub-optimal test experience to our users. We will consider "experiments" with the number of test buckets between 1 and 10. In practice, most of the time there will be no more than 5 or 6 test buckets but we would like to consider as many use cases as possible in this study. Lastly, traditional A/A validation is carried out for these "experiments" and we are interested in the resulting failure rate.

A. Simulation Procedure

Create pool of mainstream hash values with specified $tailDiscard$, according to ready-to-use algorithm.

```
$H \leftarrow$ number of mainstream hash values
for $k \in \{1, \ldots, 10\}$ do                ▷ number of test buckets
    repeat
        $m \leftarrow \infty$                    ▷ size of the control bucket
        $n \leftarrow \infty$                    ▷ size of all the test buckets
        while $m + n * k > H$ do
                         ▷ buckets must fit in the available space
            Sample $m$ from $\mathcal{U}\{10, \ldots, 200\}$
                         ▷ control bucket between 1% and 20%
            Sample $n$ from $\mathcal{U}\{10, \ldots, m\}$
                         ▷ test bucket is not larger than control
        end while
        From the mainstream pool of hash values (with
    associated cookies), randomly assign $m$ to be the control
    bucket, $n * k$ to be the $k$ test buckets.
        for $i \in \{1, \ldots, k\}$ do
            Conduct the following hypothesis tests be-
    tween the control bucket and test bucket $i$ and record the
    p-values:
        1) 2-sample $t$-test for equality of means of PV/cookie
        2) 2-sample $t$-test for equality of means of Ses-
           sions/cookie
        3) 2-sample $t$-test for equality of means of Days vis-
           ited/cookie
        4) $\chi^2$ test to check that cookies are falling into the
           buckets according to the specified probabilities
        end for
    until 1000 experiments have been completed
    Compute traditional A/A failure rate for these 1000
experiments
end for
```

B. Multiple Testing

The traditional A/A failure rate can be calculated in a few different ways. The main consideration is how to adjust for multiple testing.

When ready-to-use buckets are assigned in an experiment created on our internal experimentation platform, one control bucket is given together with the number of test buckets requested. We would like this entire experiment to pass A/A validation. That is, each test bucket should not fail any of the four hypothesis tests conducted when compared to the control bucket. This means that all $4k$ null hypotheses should not be rejected.

For a single hypothesis test, we typically fix the Type I error rate at 5% by rejecting the null hypothesis when the resulting p-value is less than 0.05. When multiple hypothesis tests are carried out together, as in a single experiment, we would like to control the family-wise error rate at 5%. The literature on correcting for multiple hypothesis testing is well established [14]–[17]. For our purposes, we employ two commonly used methods: Bonferroni correction [18], [19] and the Benjamini-Hochberg (BH) procedure for controlling the False Discovery Rate (FDR) [20].

C. Failure rate

Table II compares a few traditional A/A failure rates by the number of test buckets k under two scenarios: i) when we remove 25 hash values from each of the upper and lower tails (50 hash values or 5% in total) ii) when we remove 50 hash values (100 hash values, 10% total).

The three types of failure rates displayed are:

- "uncorrected" counts the number of experiments with any hypothesis tests with p-values < 0.05.
- "bonferroni" counts the number of experiments with any hypothesis tests with p-values $< 0.05/4k$.
- "FDR" counts the number of experiments that fail after controlling the family-wise error rate at 0.05 using the BH procedure.

One clear trend in Table II is that as the number of test buckets k increases, the uncorrected failure rate balloons from 4% for $k = 1$ to over 30% for $k = 10$. This is precisely the multiple testing problem described earlier.

We can also see that the failure rates are lower when we remove 50 hash values in the tail compared to removing 25. This is especially clear for the uncorrected rate but also true for both the Bonferroni correction and FDR.

The observed difference in failure rates under simulation between removing 25 and removing 50 is consistent with our expectations. The larger amount of the lower/upper tails removed, the more homogeneous the mainstream pool is. Therefore, when buckets are created at random from this resulting pool, they tend to be situated more closely in the metric space and less likely to fail traditional A/A validation. Suppressing A/A failures is exactly the aim of the ready-to-use A/A algorithm so we see that it is working as intended.

Table II shows that removing 25 hash values, the A/A failure rate can be as high as 1.5% while removing 50 gives us a 0.5% failure rate at most. The number of hash values to remove should be thought of as a configurable parameter according to tolerance of A/A failures. The simulation procedure above explicitly parameterizes this number with $tailDiscard$ while the ready-to-use algorithm in Section II implicitly sets it to 50.

For experimentation at Oath, 1.5% ready-to-use A/A failure rate is too high. Under traditional A/A validation, the canonical failure rate is 5% if we use an $\alpha = 0.05$ significance level and control for multiple testing. We have seen that in practice, this failure rate can be as high as 30% on real world experimentation systems. However, buckets that do fail in this way do not go on to the A/B phase. Under the ready-to-use A/A framework, there is only an A/B phase where all the buckets are used for actual A/B testing. A lower failure rate is necessary because it is undesirable for any buckets that would have failed traditional A/A validation to be used in a real experiment. After careful consideration, we decided on a 0.5% failure rate as the threshold, which is met when we remove 50 hash values in each tail.

Table II
TRADITIONAL A/A FAILURES UNDER SIMULATION

	Uncorrected (%)		Bonferroni correction (%)		FDR failure rate (%)	
k	Remove 25	Remove 50	Remove 25	Remove 50	Remove 25	Remove 50
1	4	3.8	0	0.5	0	0.5
2	13.5	7.4	0.5	0.1	0.5	0.1
3	17.5	9.6	0.5	0.3	0.5	0.3
4	27	13.5	0	0.2	0	0.2
5	28.5	18.7	1.5	0	1.5	0
6	31	21.7	1.0	0.2	1.0	0.2
7	34.5	21.5	0	0	0	0
8	40	25	0	0.2	0	0.2
9	36	27.8	0	0	0.5	0
10	44.5	29.9	0.5	0.2	1.0	0.2

There is the option of removing a lot more hash values, e.g. 200 in each tail, which would guarantee an infinitesimally small A/A failure rate. However, the space available for experimentation in each layer would decrease so that only a few very small buckets are possible, or even to the point where the entire layer is discarded. The tradeoff between a lower A/A failure rate and greater space available on the layer must be evaluated.

V. ENGINEERING IMPLEMENTATION

In this section we will outline how the ready-to-use A/A procedure is implemented in our data platform.

The use of A/B testing has grown exponentially at our company. Our multi-layer experimentation platform hosts hundreds of active layers and experiments and more than a thousand buckets on any given day. The challenge is how to enable ready-to-use A/A buckets with minimum overhead at web scale. As shown in Fig. 2, a dedicated data pipeline is devised to handle sampling and computation offline, separately from our online experimentation platform. The process includes the following four steps:

1) **Layer creation**: Layers are first created by experiment owners via the user interface (UI). As described in Section II-B, one layer consists of one thousand hash values ranging from 0 to 999. A unique seed for hash function is assigned to each layer. Layers can also be automatically created by the experimentation platform when necessary.
2) **Offline hashing and metric computation**: For each layer, user-level page views, sessions and days visited are computed based on the last seven days of historical data. We use an offline Bob Jenkins hash function with the same seed of the corresponding layer to assign users to hash values ranging from 0 to 999. The assignment by offline hashing reproduces what happens in online traffic splitting when ready-to-use A/A buckets are produced. For each hash value, mean page views, sessions and days visited per user along with sample size are computed.
3) **Hash value validation**: For a given layer, we rank the hash values by each of the four metrics: per user page views, sessions, days visited and sample size. Hash values falling into the top 50 or bottom 50 segments in any of these four ranks are excluded. The remaining ones form the pool of hash values of high quality available for ready-to use A/A bucket allocation.
4) **Experiment creation**: Experiment owners may request ready-to-use A/A buckets when they create their experiments on the UI. These buckets are certified to be sufficiently balanced without the need for traditional A/A validation.

The aforementioned daily process involves scanning through historical data at a scale of 100 terabytes. It takes about 40 minutes end-to-end to produce ready-to-use A/A buckets on all the layers in our platform. In order to achieve this, this data pipeline uses Map Reduce framework across hundreds of servers in a Hadoop cluster. Experiment owners can visualize the certification status of their layers showing space available for experimentation as shown in Fig. 3.

VI. ONLINE EVALUATION

Finally, after implementing ready-to-use A/A in our experimentation platform, we would like to examine the real performance of this framework. Similar to Section IV, we are interested in the A/A failure rate. In contrast to Section IV, the data in this section relies on online hashing by our experimentation platform. Instead of simulating bucket creation in experiments, we will simply set up a live experiment with a few buckets and observe the results of A/A validation over time.

For this study, we used a different product with a different volume of traffic from that in Section IV in order to ensure that the framework is effective when deployed across our entire suite of products. Six 4% buckets were created, with the first designated the control bucket and the other five test

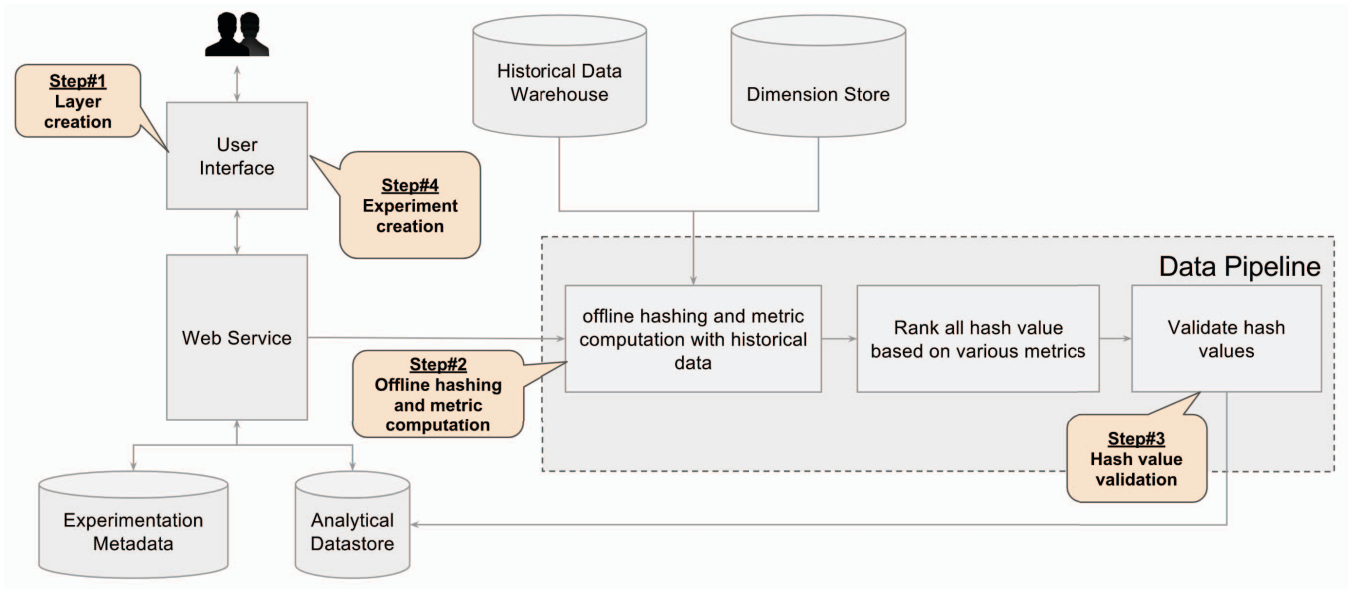
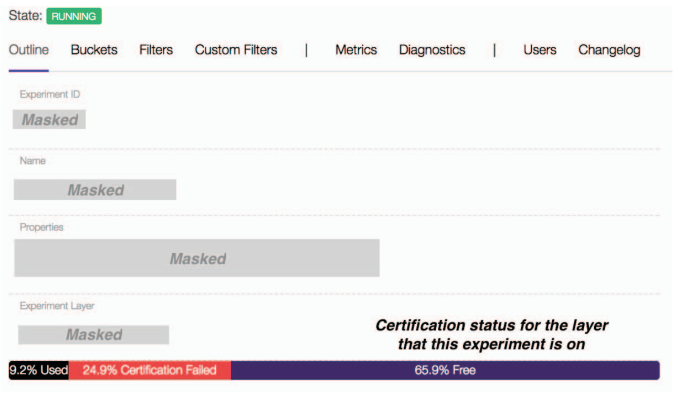


Figure 2. Offline data pipeline for ready-to-use AA

buckets. The platform backend handled the formation of the pool of mainstream hash values and random assignment to the buckets requested. The buckets are always kept in the identical control experience. After 7 days of online-stamped data has been collected, we are ready to start the evaluation, which will continue until we have 14 full days of data.

From Day 7 to Day 14, we look at data from the preceding 7 days to conduct A/A validation in a pairwise fashion for each test bucket against the designated control bucket. The 4 metrics considered are PV/cookie, Sessions/cookie, Days visited/cookie and sample size. This is the same methodology used in Section IV, with an additional time dimension to investigate the stability of our A/A certification.

Across this whole time period with five test buckets and four metrics, $8 \times 5 \times 4 = 160$ hypothesis tests were carried out. Of the resulting 160 p-values, none were below 0.05 so the failure rate, whether uncorrected, Bonferroni or FDR, is 0% throughout. If we had in fact used these six buckets for a two-week long A/B test, the data can answer the following question: *Is the observed difference in metrics between the test and control solely the treatment effect or did the buckets become unbalanced at some point in the experiment period?*


State: RUNNING

Outline Buckets Filters Custom Filters | Metrics Diagnostics | Users Changelog

Experiment ID

Masked

Name

Masked

Properties

Masked

Experiment Layer

Masked

Certification status for the layer that this experiment is on

9.2% Used 24.9% Certification Failed 65.9% Free

Figure 3. Experimentation management UI showing certified layer space

Since the A/A failure rate was 0% throughout the experiment period, this means that we maintained A/A balance for the whole duration of the test. Therefore, we can confidently say that *the observed difference is due solely to the treatment* and with appropriate, rigorous statistical analysis, we can obtain an unbiased estimate of the treatment effect.

VII. Conclusions and Future Work

A/B testing is frequently touted as the gold standard in establishing causal relationships between product features and observed changes in metrics. In order for online experiments to live up to this gold standard, A/A validation is an essential preliminary step, ensuring there is no pre-existing difference between the test and control groups. This paper presents a conceptual framework where buckets are assigned in such a way that guarantees A/A balance with high probability, thereby allowing experiments to proceed directly to the A/B phase where actionable insights can be derived. Eliminating the need for A/A validation to be explicitly carried out represents significant savings of both time and effort or resources. In addition, we have given an overview of our implementation of the proposed methodology on our internal experimentation platform. The implementation allowed us to closely examine the performance of ready-to-use A/A buckets and show that we can achieve an A/A failure rate of 0% that holds throughout a two-week long experiment.

We expect that our methodology of creating a mainstream pool of users to experiment on will be applicable to many other experimentation systems. The overview of our engineering architecture should also be helpful for those wishing to implement the ready-to-use A/A methodology in a scalable and reliable fashion.

There are a few ways our work can be further developed:

- Extend both the methodology and implementation to login ID-based experiments and Apps
- Explore whether the discarded tail segments can still be used in some way for experimentation
- Enhance monitoring to detect cases where offline hashing does not match online traffic splitting, e.g. when experimental dark matter exists
- Think about how we can infer that a bad bucket was assigned to an experiment and detect this rare occurrence. One possibility is to look back in time to the period before the experiment and check the engagement metrics of the cookies involved in the experiment. This approach is incomplete because new users are not captured but it can be a useful complement.

Acknowledgment

The authors would like to thank the Yahoo Finance and Front Page product teams for their help in setting up layers and experiments for testing, Niru Appikatala for her offline hashing package, Sudhir Chauhan for his UI work on the platform and William Choi for insightful discussion.

References

[1] R. Kohavi, T. Crook, R. Longbotham, B. Frasca, R. Henne, J. L. Ferres, and T. Melamed, "Online experimentation at microsoft," in *Proceedings of the Third International Workshop on Data Mining Case Studies, held at the Fifteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining in Paris, France*, 2009, pp. 11–23.

[2] D. Tang, "Experimentation at google," *RecSys'14 Workshop: Controlled Experimentation*, 2014.

[3] B. Frasca, "A brief history of bing a/b," *RecSys'14 Workshop: Controlled Experimentation*, 2014.

[4] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne, "Controlled experiments on the web: survey and practical guide," *Data mining and knowledge discovery*, vol. 18, no. 1, pp. 140–181, 2009.

[5] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann, "Online controlled experiments at large scale," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '13. New York, NY, USA: ACM, 2013, pp. 1168–1176. [Online]. Available: http://doi.acm.org/10.1145/2487575.2488217

[6] R. Kohavi, A. Deng, R. Longbotham, and Y. Xu, "Seven rules of thumb for web site experimenters," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 1857–1866.

[7] T. Crook, B. Frasca, R. Kohavi, and R. Longbotham, "Seven pitfalls to avoid when running controlled experiments on the web," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 1105–1114.

[8] R. Kohavi and R. Longbotham, "Unexpected results in online controlled experiments," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 2, pp. 31–35, 2011.

[9] R. Kohavi, A. Deng, B. Frasca, R. Longbotham, T. Walker, and Y. Xu, "Trustworthy online controlled experiments: Five puzzling outcomes explained," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 786–794.

[10] A. Deng, Y. Xu, R. Kohavi, and T. Walker, "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, 2013, pp. 123–132.

[11] Z. Zhao, M. Chen, D. Matheson, and M. Stone, "Online experimentation diagnosis and troubleshooting beyond aa validation," in *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 498–507.

[12] D. B. Rubin, "Estimating causal effects of treatments in randomized and nonrandomized studies." *Journal of educational Psychology*, vol. 66, no. 5, p. 688, 1974.

[13] D. Tang, A. Agarwal, D. O'Brien, and M. Meyer, "Overlapping experiment infrastructure: More, better, faster experimentation," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 17–26.

[14] P. C. O'Brien and T. R. Fleming, "A multiple testing procedure for clinical trials," *Biometrics*, pp. 549–556, 1979.

[15] Y. Benjamini and D. Yekutieli, "The control of the false discovery rate in multiple testing under dependency," *Annals of statistics*, pp. 1165–1188, 2001.

[16] R. Bender and S. Lange, "Adjusting for multiple testingwhen and how?" *Journal of clinical epidemiology*, vol. 54, no. 4, pp. 343–349, 2001.

[17] M. Aickin and H. Gensler, "Adjusting for multiple testing when reporting research results: the bonferroni vs holm methods." *American journal of public health*, vol. 86, no. 5, pp. 726–728, 1996.

[18] Wikipedia, "Bonferroni correction — Wikipedia, the free encyclopedia," http://en.wikipedia.org/w/index.php?title=Bonferroni\%20correction&oldid=773020711, 2017, [Online; accessed 05-June-2017].

[19] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961.

[20] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *Journal of the royal statistical society. Series B (Methodological)*, pp. 289–300, 1995.