Ensure A/B Test Quality at Scale with Automated Randomization Validation and Sample Ratio Mismatch Detection

Keyu Nie*
Zezhong Zhang✉*
keyunie@gmail.com
zezzhang@ebay.com
eBay, Inc.
San Jose, California, USA

Bingquan Xu
bingquanx@gmail.com
eBay, Inc.
Shanghai, China

Tao Yuan
teyuan@ebay.com
eBay, Inc.
San Jose, California, USA

ABSTRACT

eBay's experimentation platform runs hundreds of A/B tests on any given day. The platform integrates with the tracking infrastructure and customer experience servers, provides the sampling service for experiments, and has the responsibility to monitor the progress of each A/B test. There are many challenges especially when it is required to ensure experiment quality at the large scale. We discuss two automated test quality monitoring processes and methodologies, namely randomization validation using population stability index (PSI) and sample ratio mismatch (a.k.a. sample delta) detection using sequential analysis. The automated processes assist the experimentation platform to run high quality and trustworthy tests not only effectively on a large scale, but also efficiently by minimizing false positive monitoring alarms to experimenters.

CCS CONCEPTS

• **Mathematics of computing → Hypothesis testing and confidence interval computation**; • **General and reference → Experimentation**.

KEYWORDS

A/B test, experimentation, randomization validation, sample ratio mismatch, goodness-of-fit test, sequential test

ACM Reference Format:
Keyu Nie, Zezhong Zhang✉, Bingquan Xu, and Tao Yuan. 2022. Ensure A/B Test Quality at Scale with Automated Randomization Validation and Sample Ratio Mismatch Detection. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3511808.3557087

1 INTRODUCTION

A trustworthy experimentation platform is the foundation to support eBay's data driven business and product decision making. A/B tests running on the platform are based on the randomized controlled trial (RCT) [6, 7] that is widely regarded as the gold standard of causal inference. However, in the industry, such as Google [7], Meta, Microsoft [3, 6, 7], Linkedin [1], Twitter, Yahoo [23] and others, studies with randomization quality related concerns are commonly reported. These quality related issues are flawed and harmful because they can lead to incorrect analysis and conclusion.

In practice, there are two types of randomization issues that draw our attention. The first is imperfect randomization, which occurs during the traffic/sample randomization assignment. Consider a platform provides a randomization engine that divides web traffic into small buckets evenly (say 100 buckets, so each bucket represents 1% out of the total traffic). If one of the bucket had more than 1% traffic (for example, 1.1% actual traffic or 10% relative deviation), we would suspect a problem with our randomization engine. As a result, a sample randomization validation check is critical to ensure that samples are dispersed evenly in a uniform Multinomial distribution. In fact, "unhappy randomization" and operating issues are the most common causes of faulty randomization results [12].

The other common A/B test quality issue arises from mis-matching of the expected sample ratio to the observed sample ratio between test and control groups in triggered users. For example, consider an experiment with $10,000$ users total traffic in which each user has an equal chance of experiencing test and control features, a.k.a. 50-50 split. The expected sample ratio would be 50/50=1. The observed sample ratio is 2/3 if the experiment ends with just $2,108$ (test) and $3,183$ (control) triggered users in the test and control groups respectively (similar example can be found in [9, 10]). The sample ratio mismatch between the expected and observed could also raise concerns on the validity of an experiment. It may indicate issues in the experiment implementation (as Microsoft suggests in [3]), e.g., errors in the event tracking, user experience service programming code, and bot filtering-related data processing.

These errors are usually beyond the control of experimentation platform. However it is the role of the experimentation platform to monitor and send accurate alerts so that experimenters can investigate on the ramification of the situation as soon as possible. While randomization validation and sample ratio mismatch detection are used to monitor the quality of A/B tests, the majority of these alerting approaches themselves are statistical hypothesis testings. The main challenge of the alerting is to balance between the essential need of early detection of quality issues (sensitivity) and the minimization of false alarms (due to the nature of randomness).

Standard goodness-of-fit tests, such as Pearson χ^2 test [2, 17], Kolmogorov–Smirnov test (KS test) [11] and Anderson-Darling test

*Both authors contributed equally to this research.

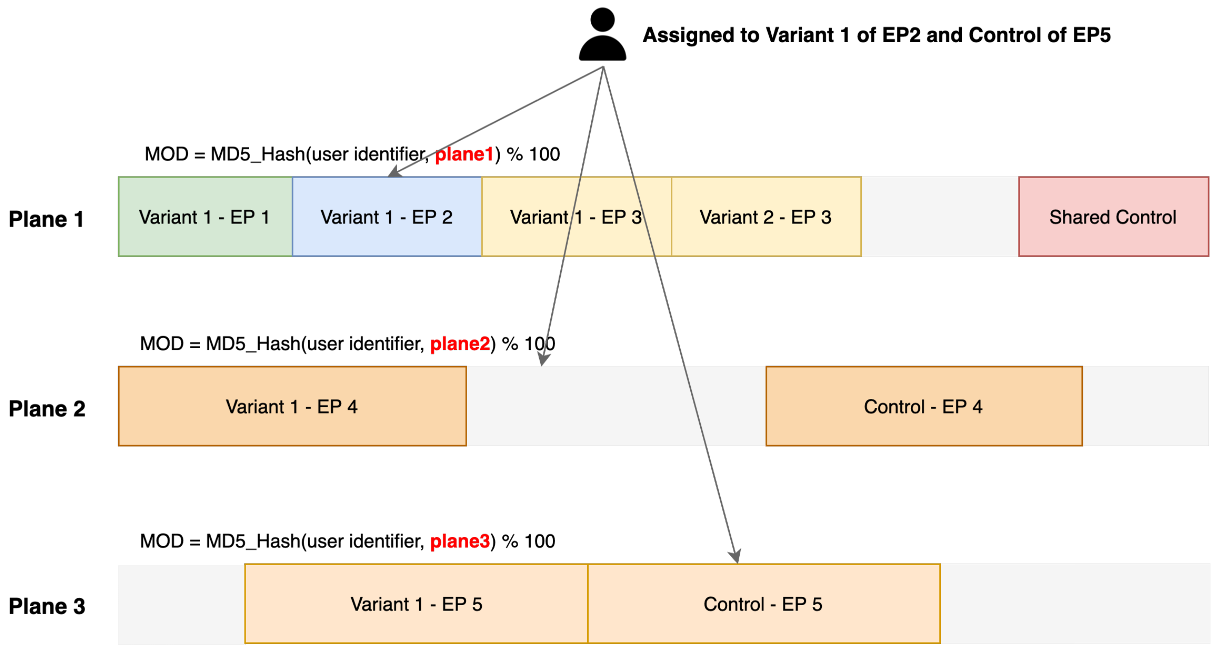
Assigned to Variant 1 of EP2 and Control of EP5
MOD = MD5_Hash(user identifier, plane1) % 100
Plane 1
Variant 1 - EP 1
Variant 1 - EP 2
Variant 1 - EP 3
Variant 2 - EP 3
Shared Control
MOD = MD5_Hash(user identifier, plane2) % 100
Plane 2
Variant 1 - EP 4
Control - EP 4
MOD = MD5_Hash(user identifier, plane3) % 100
Plane 3
Variant 1 - EP 5
Control - EP 5

Figure 1: Exclusive and Orthogonal Tests (Each plane id is associated with a distinct random value.)

(AD test) [15, 16] provided in Google, Microsoft and Linkedin [7], are common solutions for randomization validation of uniformly Multinomial distributions. According to Yahoo [23], Linkedin [1] and Microsoft [3], the Pearson χ^2 test and two sample t-test are utilized in sample ratio mismatch detection. Our initial version of sample ratio mismatch detection algorithm was based on a t-test that was overly sensitive and generated tons of noisy signals. For example, the observed sample ratio of 2/3 might be OK at the early stage with only 2 and 3 users in each group, but it would worry us if the sample size was extended to 2000 and 3000 users, respectively. Our current algorithm is built on a series of adaptive sequential tests that significantly reduces false alarms while maintaining the sensitivity we need to effectively monitor sample delta for hundreds of experiments daily without spamming experiment owners.

In the paper, we describe two levels of test quality monitoring and alerting, (1) sample randomization check for traffic assignment, and (2) sample ratio mismatch detection for the triggered (or qualified) samples. Both are targeted to minimize false positives in their processes while attaining good recall of the signals. The methodologies involve a novel Population Stability Index (PSI [21]) based test and a sequential probability ratio test (SPRT [4, 5, 14, 18, 19, 24]). To our knowledge, we are the first to automate the methods (in early 2019) in a large scale experimentation platform to continuously monitor experiment quality.

The paper is structured as follows: first we briefly describe our randomization engine in section 2, then we introduce the PSI test for randomization check in section 3, it follows with our sequential probability ratio test for sample delta detection in section 4. We provide the application use cases of diagnosing test issues using the methodologies. We conclude in section 5.

2 RANDOMIZATION ARCHITECTURE

2.1 Sample Assignment

In the experimentation platform, the randomization engine leverages the standard MD5 algorithm and math modulo (MOD) function to hash samples uniformly into small buckets.

$$\big(\mathbf{Hash}(x, seed) \mod B\big) \sim \mathbf{Multinom}\big(\frac{1}{B}\big) \tag{1}$$

where x is the sample identifier (e.g., cookie id), $\mod$ is the arithmetic modulo operator. $\mathbf{Hash}(x, seed)$ is the hashing function with a randomization seed (e.g., plane-id). B is the total number of traffic buckets (e.g., 100).

The experimentation platform supports exclusive and orthogonal testing modes [1], see Figure 1. The exclusive mode allows multiple experiments running on a shared plane (by hashing with the same randomization seed). The orthogonal mode runs an independent feature experiment on its own plane (by hashing with its own randomization seed). Each plane covers the whole online traffic, and is divided into 100 uniform buckets. Buckets can be grouped into "swim lanes" (e.g., buckets with mod value from 0-49) and assigned to test and control variants. A bucket on a plane can only be assigned to one variant group of an experiment at a given time. The more buckets a variant occupies, the more traffic it is expected to have.

For an experiment, one user is assigned a mod value that corresponds to a specific bucket on a certain plane. This mod value

[1]Independent features can be tested orthgonally by using different "planes", e.g. one plane for ranking algorithm, another for UI presentation. However, if two features can cause severe collision of user experience, they must be tested exclusively by using a shared "plane".

varies from different planes. The randomization engine should distribute the user traffic uniformly and randomly into test and control buckets for an experiment. A randomization failure can cause unbalanced assignments of the samples among the buckets. When this happens, the experiment becomes untrustworthy and is extremely difficult to rectify using post-processing techniques.

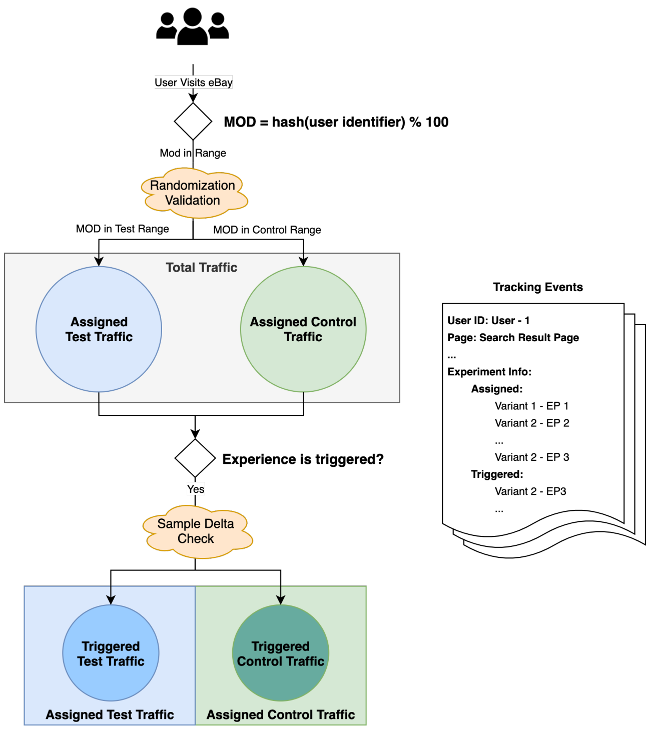

Figure 2: Experiment Sampling Process

2.2 Sample Triggering

After a sample user is assigned to a variant in an experiment by the randomization engine, the user is only considered triggered, when he or she actually visits the eBay application, see Figure 2.

Failure can occur in the sample assignment and qualification stages. A randomization failure can manifest itself in a variety of ways. Some of them are simple and straightforward to spot. For example, allocating the same user to both the test and the control is incorrect yet easy to detect. However, sample ratio mismatch, or the discrepancy between realized and expected traffic ratio between groups, can be difficult to investigate. Detecting such issues as early as possible can help experimenters to fix the underlying issues without further loss of time and resources. Such detection functions are essential for the effectiveness of running experiments at scale and ensures the trustworthiness of all A/B tests.

3 RANDOMIZATION VALIDATION WITH A POPULATION GOODNESS-OF-FIT TEST

3.1 Problem Statement

As described above, our randomization engine employs MD5 hashing and MOD functions [6] (in equation 1). It is important to check whether the distribution among traffic buckets is uniform and evenly distributed for a running experiment. We call it an "in-flight randomization validation" as it can be performed while an experiment is in progress.

Given an experiment, we denote the sample size n_b in bucket b and total sample size $n = \sum_{b=0}^{B-1} n_b$, the goal is to test whether the sample distribution $\hat{p}_b = \frac{n_b}{n}$ matches the expected distribution $\frac{1}{B}$ in Multinomial when evenly distribution is by design. The hypotheses of the distribution test is

$$\begin{aligned} \mathrm{H}_0 &: \forall b \in [0, B), p_b = \frac{1}{B} \\ \mathrm{H}_a &: \exists b \in [0, B), p_b \neq \frac{1}{B} \end{aligned} \tag{2}$$

To validate the randomization results while collecting samples, we can utilize industry standard statistical goodness-of-fit tests [1, 7] as baselines, e.g., Pearson χ^2 test [2, 17], KS test [11] and AD test [15, 16]. The drawback of the widely adopted methods is that they generate non-ignorable false positives and false negatives that impact the effectiveness of the randomization validation process, i.e., either missing signals, or generating many false alerts.

Considering that our platform can run hundreds and thousands of experiments concurrently, an algorithm with a 0.5% false positive rate may slow down the test pace for unnecessary manual investigation, resulting in delay and loss of product development opportunity. In addition to the standard algorithms, we developed a new method based on Population Stability Index (PSI [21]).

3.2 PSI and PSI$_k$ Tests

Population Stability Index (a.k.a. PSI) is computed as

$$\mathbf{PSI} = \sum_{b=0}^{B-1} (\hat{p}_b - \hat{q}_b) \ln \frac{\hat{p}_b}{\hat{q}_b} \tag{3}$$

where $\hat{p}_b$ and $\hat{q}_b$ are the sample proportions in the bucket b (with total B buckets) from two distributions $\mathbf{Multinom}(\mathbf{p})$ and $\mathbf{Multinom}(\mathbf{q})$.

PSI is a test statistic [21] that has been used to measure how much the distribution of a random variable has shifted over time, or to measure the distribution difference. Therefore, the original PSI is a two-sample hypothesis testing with m as the other total sample size from the reference distribution $\mathbf{Multinom}(\mathbf{q})$ to be compared. In theorem, $\frac{1}{\frac{1}{n}+\frac{1}{m}}\mathbf{PSI}$ approximately follows a χ^2 distribution with $B-1$ degrees of freedom[21],

$$\mathbf{PSI} \sim (\frac{1}{n} + \frac{1}{m}) * \chi^2_{B-1} \tag{4}$$

where n and m are the total sample size from two distributions $\mathbf{Multinom}(\mathbf{p})$ and $\mathbf{Multinom}(\mathbf{q})$ respectively. To distinguish the two total sample size, we use m as the reference sample size and n as the assigned sample size.

In the randomization validation case, we adapt PSI to determine if the assigned sample distribution over buckets $(n_0, \ldots, n_{B-1})$ is the same as sampling from an uniform Multinomial distribution $(\frac{kn}{B}, \ldots, \frac{kn}{B})$ with the reference total sample size $m = kn$. Here $k \in \mathbb{N}^+$ is the ratio of the reference sample size to the assigned total sample size. We found that k could be regarded as a hyper parameter, and our algorithm performed slight better by setting $k = 2$ as a practice trick.

From Equation 4, we can test whether the assigned sample distribution $\hat{p}_b = \frac{n_b}{n}$ are evenly distributed across B buckets ($\hat{q}_b = \frac{1}{B}$) with a test statistic $\mathbf{PSI}_k$ defined as,

$$\begin{aligned} \mathbf{PSI}_k &= \sum_{b=0}^{B-1} (\frac{n_b}{n} - \frac{1}{B})(\ln \frac{n_b}{n} - \ln \frac{1}{B}) \\ &\sim \frac{k+1}{kn} \chi^2_{B-1} \end{aligned} \quad (5)$$

where $k \in \mathbb{N}^+$ is a hyper parameter and can be interpreted as the ratio of the reference sample size to the assigned sample size n.

We can construct a decision rule along with significance level α for sending imperfect randomization alert when

$$\mathbf{PSI}_k > \frac{k+1}{kn} \chi^2_{\alpha,B-1} \quad (6)$$

where $\chi^2_{\alpha,B-1}$ is the critical value that has $(1-\alpha) \times 100\%$ area from the left tail of χ^2_{B-1} distribution with $B-1$ number of degrees of freedom. As a trival example, when $k = 1$, the decision region will be $\mathbf{PSI}_{k=1} > \frac{2}{n} \chi^2_{\alpha,B-1}$.

Our precision and recall analysis and comparison in the following demonstrates that $\mathbf{PSI}_k$ performs the best both in our real-data and simulations. It is robust and can dramatically reduce false positive alarms in our production pipeline, which motivates us to share our findings with broader audience.

3.3 Performance Analysis

The non-ignorable false positive rate is the fundamental concern with standard goodness-of-fit testings. We show that **PSI** can further suppress false positives when compared to KS test, the best randomization validation test so far in industry. We constructed three datasets from the real production data and simulation to prove our point, as summarized in Table 1. To evaluate the performance, we reported the false positive rate (FPR), precision, recall and F-score on three datasets.

Dataset	Description
1	306 negative cases collected in real data
2	291 negative cases collected in real data 33 positive cases with on average 0.09% noise
3	500 negative (evenly distributed) 100 positive cases with on average 0.09% noise simulated based on Multinomial distribution

Table 1: Datasets for Randomization Validation Comparison

The Dataset 1 was collected from real online experiment metadata which contains no sample distribution anomaly, and all cases are considered negative.

The Dataset 2 was based on the experiment metadata with no anomaly but randomly mixed with sample count anomalies, causing the buckets to be imbalanced. Specifically, we artificially added an extra $(0.05 + x10^{-2})\%$ of sample counts into certain bucket, where $x \sim Poisson(4)$. By doing so, we injected on average 0.09% noise into certain buckets as the imbalanced cases which is aligned with our daily practice. This resulted in small but positive sample count distribution anomaly due to extra samples in buckets. In a summary, we generated 33 positive cases and kept 291 negative cases. For Dataset 1 and 2, the total sample size of collected experiments ranged from half million to several billion samples, which includes both small and big historical experiments in eBay.

The Dataset 3 is coming from simulation with negative and positive cases based on Multinomial distributions purely for reproduction purpose. Specifically, the sample count of a bucket is drawn from a Multinomial distribution with $p_b = \frac{1}{100}$, and total sample size $n \sim Poisson(3 \times 10^6)$. Similar to Dataset 2, we randomly picked up to 5 buckets to inject anomalies with new sample proportions in these buckets being $p_b = \frac{1}{B} + (0.05 + x10^{-2})\%$ where $x \sim Poisson(4)$, which injects on average 0.09% noise. In total, we simulated 500 negative and 100 positive cases.

We found that, in Table 2, AD test performed better than Pearson chi-square test, while KS test was the best out of the three. But there was still 0.33% FPR for KS test. It translates to 3 alerts among 1000 concurrent experiments at any given time, which is barely acceptable. $\mathbf{PSI}_{k=1}$ had zero FPR in Dataset 1. This might be a coincidence, but it also suggested $\mathbf{PSI}_{k=1}$ could further silence false alerts comparing with KS test.

Dataset 1	χ^2 test		AD test		KS test		$\mathbf{PSI}_1$ test	
	0	1	0	1	0	1	0	1
True Label 0	275	31	302	4	305	1	306	0
False Positive Rate	10.13%		1.31%		0.33%		0%	

Table 2: PSI performance evaluation on Dataset 1.

Table 3 also proved that the $\mathbf{PSI}_{k=1}$ outperformed all baselines with the highest F-score 98.46%. This is a significant achievement in practice, since our platform is expect to run thousands of experiments concurrently, even an 1% improvement in reducing FPR can help us avoid unnecessary investigations.

Dataset 2	χ^2 test		AD test		KS test		$\mathbf{PSI}_1$ test	
	0	1	0	1	0	1	0	1
True Label 0	261	30	287	4	290	1	291	0
True Label 1	0	33	3	30	3	30	1	32
FPR (in %)	10.31%		1.37%		0.34%		0%	
Precision (in %)	52.38%		88.24%		96.77%		100%	
Recall (in %)	100%		90.11%		90.91%		96.97%	
F-score (in %)	68.75%		89.55%		93.75%		98.46%	

Table 3: PSI performance evaluation on Dataset 2.

Table 4 showed that **PSI** performed similarly in simulated Dataset 3. $\mathbf{PSI}_{k=1}$ testing was found to be sensitive and accurate in both precision and recall measures. Readers are welcome to reproduce the results by themselves.

Dataset 3	χ^2 test		AD test		KS test		$\mathbf{PSI}_1$ test	
	0	1	0	1	0	1	0	1
True Label 0	447	53	494	6	496	4	500	0
True Label 1	0	100	22	78	27	73	0	100
FPR (in %)	10.60%		1.20%		0.80%		0%	
Precision (in %)	65.36%		92.86%		94.81%		100%	
Recall (in %)	100%		78.00%		73.00%		100%	
F-score (in %)	79.05%		84.78%		82.49%		100%	

Table 4: PSI performance evaluation on Dataset 3.

Dataset	$\mathbf{PSI}_k$ test	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$
Dataset 1	False Positive Rate	0.00%	0.00%	0.00%	0.00%	0.33%	0.33%	0.98%
Dataset 2	False Positive Rate	0.00%	0.00%	0.00%	0.00%	0.34%	0.34%	1.03%
	Precision	100%	100%	100%	100%	97.06%	97.06%	91.67%
	Recall	96.97%	100%	100%	100%	100%	100%	100%
Dataset 3	False Positive Rate	0.00%	0.00%	0.00%	0.00%	0.00%	0.20%	0.60%
	Precision	100%	100%	100%	100%	100%	99.01%	97.09%
	Recall	100%	100%	100%	100%	100%	100%	100%

Table 5: $\mathbf{PSI}_k$ performance evaluation varying $k = 1, 2, 3, 4, 5, 6, 7$

The noise scale 0.09% is discussed according to practice, while our proposed algorithm actually perform very well under different noise scale. In Appendix B, we report the performance under different noise scale from 0.05% to 0.15% on average.

As a trick in practice, we could fine-tune the performance of $\mathbf{PSI}_k$ by varying with different k values for different use cases. Table 5 demonstrated an example of k tuning using grid search over k and summarized the results in our datasets. We discovered that, by increasing k, we may have higher recall while could damage precision. We finally settled with $k = 2$ in eBay according to the human diagnose feedback of detected experiments. While, it is still an interesting open question on tuning the k from the data.

In a summary, $\mathbf{PSI}_k$ is a viable solution along with other well-known methods with on-par or better performance on precision and recall. It will be interesting for other researchers to confirm its performance on other data sets. $\mathbf{PSI}_k$ is an indispensible tool to improve the effectiveness of our randomization engine.

3.4 Randomization Case Study

We have implemented $\mathbf{PSI}_k$ in our experimentation platform for the randomization "in-flight" validation monitoring. In practice, we occasionally observe imperfect randomization results (samples are not evenly distributed among buckets) due to "unhappy randomization" and operation issues. A operation issue on experience tracking and randomization engine is severe as it could corrupt more than one experiment. After our deployment of $\mathbf{PSI}_k$, we can catch the imperfect randomization issues at a very early stage to take actions, instead of dealing with experimenters' complains at the end of data collection. To be honest, issue from the randomization engine is usually rare. Below, we share two cases here, in one case it helped us catch a bug.

Case 1. Sample Leakage Detection.

Experiment: In 2021, one domain team ran an experiment with 10%-10% traffic split. The samples with mod values 0-9 were assigned to the control and 10-19 to the test group in the experiment setting, while the rest samples with mod values 20-99 are unassigned traffic buckets. Shortly after our $\mathbf{PSI}_k$ deployment, it caught an imbalanced sample count among the 0-99 buckets in this experiment. Moreover, there exists more samples in each bucket with mod values 0-19 than 20-99, where some "ghost" users accidentally leaked into two or more buckets. In the triggered sample report, we observed about 0.3% of "ghost" users with mod values 20-99 appeared in either test or control group. Interestingly, these "ghost" users were uniformly dispatched among the unassigned traffic buckets.

Explanation: After further research, we concluded that the incident was not caused by a flaw in our analytic data pipeline, but rather by a malfunction on the experience service side. A tracking application recorded incorrect user ids (cookies) at a specific point in the service pool. The cookie ids recorded in the logs were different from those fed into the randomization engine. This could happen as the application wrongly decide which user ids (cookies) to track the user given the situation at a specific point and time during the experiment period. It appeared as if some users accidentally leaked into different randomization buckets in an experiment, and caused a randomization anomaly. $\mathbf{PSI}_k$ monitoring was able to successfully detect the sample imbalance early to initiate an investigation. It is important to note that if one of the buckets belongs to the control and another belongs to the test group, it may trigger sample delta alert on the triggered samples which we discuss in the next section. While such mistakes could pollute several experiments together, it is important to alarm earlier on total traffic assignment instead of waiting until multiple experiments triggered sample delta alert.

Case 2. Unhappy Randomization.

Because the hashing and MOD functions aren't perfect, the randomization output will not always meet an experiment's requirement. Researchers have discovered that choosing the proper randomization seed reduces the risk of sample size differences. Re-randomization is a technique for reducing the noise produced by "unhappy randomization" and improving the test precision [12]. As $\mathbf{PSI}_k$ can identify unhappy randomization early, we may warn experimenters to re-randomize samples when seeing imbalanced buckets using a different randomization seed.

4 SAMPLE RATIO MIS-MATCH MONITORING

4.1 Delta among Triggered Sample Counts

Imagine in an experiment, 50 buckets on a plane are assigned to the test group and 50 buckets to the control group. The test and control groups should have roughly the same number of samples. However we may still observe a noticeable or significant difference between the observed sample count from the expected value, e.g. there may be 1% more samples in the test group than that of the control group. In our experience, a 1% deviation is more surprising with large samples but may be expected on small samples. We call this “sample delta” or sample ratio mismatch in the literature.

Although a small difference is always expected due to tracking data time delay and traffic fluctuation, the magnitude of sample delta issues can raise concerns. We developed a sample delta detection method based on sequential analysis with a significance threshold to monitor all live experiments. It becomes one of the most critical health checks in our experimentation platform. For both the assigned traffic and triggered traffic of each experiment, we calculate sample delta respectively for diagnostic and validation purposes. As the method can be applied in both assigned (randomization only) and triggered sample sets, we use the triggered samples to illustrate the principle of the method.

4.2 Sample Delta Sequential Test

Let $x_1^T, \ldots, x_k^T$ and $x_1^C, \ldots, x_k^C$ be the cumulative sample count in the control (C) and treatment (T) groups of an experiment at day k with assigned traffic r^T and r^C. We can construct a two-sided hypothesis test under binomial setting with observed sample ratio $p = \frac{x_k^T}{x_k^T + x_k^C}$ and expected sample ratio $p_0 = \frac{r^T}{r^T + r^C}$:

$$H_0 : p - p_0 = 0$$
$$H_A : p - p_0 \neq 0$$

The naive design hinges on a proportion metric illustrated in the above hypothesis. As proportional t-tests are usually very sensitive, the hope is to catch the sample delta as soon as possible. In fact, it turns out the t-test is too sensitive especially when the sample size $n_k = x_k^T + x_k^C$ is large, as it will almost certainly detect a small but negligible difference (e.g. 0.1%) throughout the days of an experiment with millions of samples. But a 0.1% deviation with millions samples is very common in our daily practice and should not raise concern on our system. In the meanwhile, we still expect to get alert for big sample delta (e.g. 1%) with moderate sample size.

So we change our sample delta detection test hypothesis to include a threshold as following:

$$H_0 : |p - p_0| \leq \delta$$
$$H_A : |p - p_0| \geq \delta$$

where δ is an error tolerance hyper parameter that can be tuned to surpress the noises. To start with, we set $\delta = \min(1\%, 5\% min(p_0, 1 - p_0))$.

Since we are going to run the same sample delta test repeatedly for each experiment at an hourly or daily basis, it would definitely inflate the false positive (type-I error) rate for the detection. Instead of conducting a proportional t-test, we design the detection based on the sequential probability ratio test (SPRT) [18] to minimize the inflated type-I errors. Specifically, we design two adaptive SPRT tests (one-sided):

$$\begin{array}{ll} TestA & TestB \\ H_0 : p - p_0 = 0 & H_0 : p_0 - p = 0 \\ H_A : p - p_0 \geq \delta & H_A : p_0 - p \geq \delta \end{array}$$

with the corresponding test statistics (see Appendix A.1) as:

$$t_k^A = \log \frac{l_{H_A}}{l_{H_0}} = -\frac{\delta^2 - 2\delta(\hat{p}_k - p_0)}{2\hat{\sigma}_k^2} \quad (7)$$

$$t_k^B = \log \frac{l_{H_A}}{l_{H_0}} = -\frac{\delta^2 + 2\delta(\hat{p}_k - p_0)}{2\hat{\sigma}_k^2} \quad (8)$$

where $\hat{p}_k = \frac{x_k^T}{x_k^T + x_k^C}$ (the symbol $\hat{*}$ means it is computed in a real example) and $\hat{\sigma}_k^2 = \frac{\hat{p}_k(1-\hat{p}_k)}{n_k}$. A more accurate version of the formula (SPRT-EXACT) using binomial distribution directly instead of the normal distribution via CLT (see Appendix A.2) can be:

$$t_k^A = x_k^T log(\frac{p_0 + \delta}{p_0}) + x_k^C log(\frac{1 - p_0 - \delta}{1 - p_0}) \quad (9)$$

$$t_k^B = x_k^T log(\frac{p_0 - \delta}{p_0}) + x_k^C log(\frac{1 - p_0 + \delta}{1 - p_0}) \quad (10)$$

The alerting rules of the SPRT tests are based on the critical region [18] with type-I α and type-II β error bounds. For any $k = 1, \ldots, \infty$:

- if $t_K > log(\frac{1-\beta}{\alpha})$:
  Send alert and break;
- if $t_K < log(\frac{1-\beta}{\alpha})$:
  Continue monitoring;

4.3 Performance Analysis

We choose the t-test and chi-square tests as the methodological comparison baseline as they are the most commonly used, to illustrate the performance of the SPRT. The validation dataset is based on 519 long running experiments with average K = 29 days data points.

The dataset is labeled with the criterion that if $\frac{|x_k^T - x_k^C|}{x_k^C} > 1\%$ as the evidence of sample delta existed. The label is reflected in the following table 6, where 0 indicates that there was no sample delta issue and 1 indicates that there was a sample delta issue. According to our experience, the rule may result in small amount of false positives, particularly when sample size is small, but no false negative.

We report the precision as the primary evaluation metric as well as recall in our confusion table (Table 6). We choose $\alpha = 0.05$, $\beta = 0$ for SPRT and SPRT-EXACT methods. We also intentionally lower α to 0.01 for the chi-square and t-test statistics to further reduce their family wise errors.

	χ^2 test ($\alpha = 1\%$)		t-test ($\alpha = 1\%$)		**SPRT** ($\alpha = 5\%$)		**SPRT-EXACT** ($\alpha = 5\%$)	
	0	1	0	1	0	1	0	1
True Label 0	13798	719	13798	719	14501	16	14501	16
True Label 1	309	285	307	283	310	282	310	282
Precision	28.39%		28.24%		**94.63%**		**94.63%**	
Recall	47.98%		47.97%		47.64%		47.64%	

Table 6: Sample delta detection method comparison

Table 6 indicates that both SPRT and SPRT-EXACT can dramatically increase alert precision from 28% to 95%. Actually SPRT and SPRT-EXACT have identical results. The SPRT and SPRT-EXACT tests appear to be superior to the direct t and chi-square tests.

It's worthy to note that the rule-based labels are used in all four methods, resulting in a 48% recall rate. We plot a bar graph of the recall rate versus sample sizes in Fig. 3. The graph illustrates that as the sample size grows, the SPRT greatly boosts the recall rate. This could be partially explained by rule-based labeling' significant false positives in small samples. We maintain that the key focus here is to the significant reduction of the type-I error when monitoring a large number of online A/B tests in an experimentation platform.

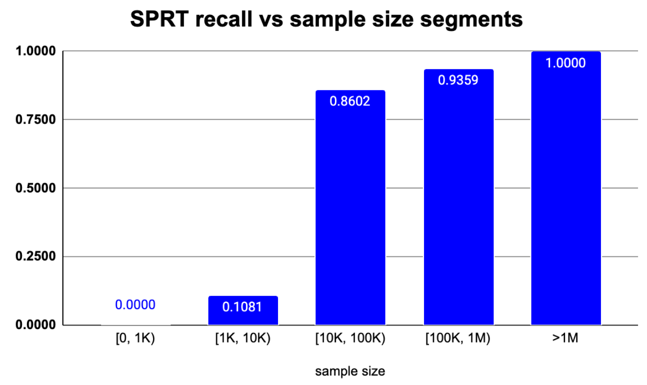

Figure 3: SPRT recall rate in different sample size segments.

4.4 Sample Delta Debugging and Case Study

Our experimentation platform implements the automated detection and monitoring process for sample delta troubleshooting. The proposed Sample Delta method has been monitoring the triggered samples and these samples under different user cohorts for each eBay experiment since early 2019. It provides insights by the following aspects whenever a sample delta is detected for an experiment:

(1) By date: did the sample delta appear only after a certain date?
(2) By the triggered sample set or assigned sample set: did the sample delta appear only in the triggered traffic or in the MOD assigned and qualified traffic as well?
(3) By the impact on other experiments: did any other experiments suffer sample delta issue on/from the same day as well? Did any experiment have any significant metric change starting from the start day of the sample delta issue?
(4) By site/channel/browser families: did the sample delta only appear on certain sites, channels(Android, iOS, Web etc.), certain types of browsers?

Information from different aspects can be combined to generate deeper insight to help troubleshoot the sample delta issue. As shown in Figure 2, the sample delta detection evaluates the cumulative sample counts at both assigned and triggered level for each live experiment by all the aspects mentioned above to see if a sample delta is significant. If the sample delta is significant for an experiment, the experiment owner must identify what the root cause is, fix the error and re-run the experiment. The faster the platform notices the problem, the earlier the experimenter can take action. Most of the sample ratio mismatch cases caused by underlying issues have large differences that don't require a significant amount of samples to detect. As a result, a real-time data pipeline has also been implemented in the platform in order to detect these issues more quickly.

The procedure of troubleshooting a sample delta issue is arduous because there can be many probable causes. Most of the root causes lie outside of the experimentation platform. The platform can assist by narrowing down the problem space to facilitate the debugging process. For example, if the sample delta appears only in the triggered traffic after a particular date but not in the assigned traffic, it may indicate a code release on the experimenter's development side, with an implementation issue propagated to the triggered traffic on that date. In the following, we review two real-life examples that we came across in practice.

Case 1. Bot Filtering of Online Traffic.

Experiment: In 2021, one domain team observed sample delta alerts at both the triggered and assigned traffics, shortly after the beginning of the experiment. The sample delta was much larger in the triggered traffic ($\approx 5\%$) than that from the assigned traffic ($\approx 3\%$). The sample delta was mostly flat ($\approx 0\%$) in many country sites except significant ($\approx 11\%$) in a particular country **F**. In the meanwhile, for many other experiments running in the particular country **F**, the total samples and the user sessions started to increase right after the problematic experiment started to run. The experimentation platform detected the sample delta ratio mismatch and alerted the experiment owners.

Explanation: After a comprehensive diagnose, we conclude that the test variant caused a latency issue on the particular country **F**'s site. This latency messed up with system bolt filtering logic and hence broke the traffic balance. Online application tracking events usually cannot be consumed directly by analytics and experiments without going through proper bot detection. Figure 4 illustrates tracking data with experimentation tags can be generated from the customer's client side and the online server side applications. It is then written into Kafka pools. The tracking data should be processed such as to filter out the external bots and the internal server side events, only genuine customer traffic are allowed to flow through to the downstream applications.

At the time, one of the common rules of bot detection would recognize a visitor as a bot if it had too many user activities in a short period of time. In this case, when the test variant slowed down the traffic, a significant amount of the bot traffic would not be recognized as bots because the activities took longer time. As such the experiment saw the imbalanced traffic between test and control due to additional un-filtered bot traffic in the test group. For other experiments, the session count and sample count also increased after the experiment started.

Case 2. Online Experience Redirect.

Experiment: In 2021, a development team rewrote their service on a new tech stack (technology migrations). Then, they ran an experiment to see the impact on metrics to compare the old with the new service. The traffic hit the old service directly as it was in production. However for a user in the test group, the experiment redirected the traffic to the new service. A sample delta at the triggered sample level was observed immediately after the experiment started but the randomization assigned traffic was flat.

Explanation: It turned out the additional hop of the redirection was the culprit. Additional breakdown on service identifier showed that the assigned traffic on the old service is flat, but the amount of assigned test traffic on the new service is much less than the assigned traffic on the old service in control. The redirection produced a non-negligible failure rate and delays, and resulted in a sample ratio mismatch alert in triggered traffic. The team re-implemeted the logic and restarted the experiment without a further issue.

5 SUMMARY AND FUTURE WORK

In the paper, we describe two methods and processes to ensure A/B test quality at the sample assignment and triggering stages for experimentation needs. The novel population stability index (PSI) test method and the sequential analysis algorithm enable the ability to automatically monitor potential randomization issues and sample discrepancies among the control and test groups for all experiments at scale with the required sensitivity but minimized false alerts. We deployed the two methods in experimentation platform for monitoring and alerting the health of every experiment in eBay. They are critical for the platform to successfully execute thousands of high quality and trustworthy experiments to promote the experimentation culture among product developers. In the future, we plan to implement other methods to continuously monitor experiment pre-existing bias in decision metrics, e.g., accumulative and ratio metrics [13]. Besides, testing immature features could cause outages or critical harmness on key business metrics. The monitoring on key metrics [20, 22, 24] can help alarm experimenter at the very early hours and ensure a safe data collection, which can be another interesting application to further explore.

REFERENCES

[1] Nanyu Chen, Min Liu, and Ya Xu. 2019. How A/B Tests Could Go Wrong: Automatic Diagnosis of Invalid Online Experiments. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 501–509.

[2] William G Cochran. 1952. The $\chi 2$ test of goodness of fit. *The Annals of mathematical statistics* (1952), 315–345.

[3] Aleksander Fabijan, Jayant Gupchup, Somit Gupta, Jeff Omhover, Wen Qin, Lukas Vermeer, and Pavel Dmitriev. 2019. Diagnosing sample ratio mismatch in online controlled experiments: a taxonomy and rules of thumb for practitioners. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2156–2164.

[4] Eugene Kharitonov, Aleksandr Vorobev, Craig Macdonald, Pavel Serdyukov, and Iadh Ounis. 2015. Sequential Testing for Early Stopping of Online Experiments. *SIGIR* (2015).

[5] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann. 2013. Online controlled experiments at large scale. *KDD* (2013).

[6] 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.

[7] Ron Kohavi, Diane Tang, and Ya Xu. 2020. *Trustworthy online controlled experiments: A practical guide to a/b testing*. Cambridge University Press.

[8] Erich Leo Lehmann, Joseph P Romano, and George Casella. 2005. *Testing statistical hypotheses*. Vol. 3. Springer.

[9] Michael Lindon. 2022. *A Better Way to Test for Sample Ratio Mismatches (SRMs) and Validate Experiment Implementations.* https://medium.com/engineers-optimizely/a-better-way-to-test-for-sample-ratio-mismatches-srms-and-validate-experiment-implementations-6da7c0d64552

[10] Michael Lindon and Alan Malek. 2020. Anytime-Valid Inference for Multinomial Count Data. https://doi.org/10.48550/ARXIV.2011.03567

[11] Frank J Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association* 46, 253 (1951), 68–78.

[12] Kari Lock Morgan and Donald B Rubin. 2012. Rerandomization to improve covariate balance in experiments. *The Annals of Statistics* 40, 2 (2012), 1263–1282.

[13] Keyu Nie, Yinfei Kong, Ted Tao Yuan, and Pauline Berry Burke. 2020. Dealing With Ratio Metrics in A/B Testing at the Presence of Intra-User Correlation and Segments. In *International Conference on Web Information Systems Engineering*. Springer, 563–577.

[14] Leo Pekelis, David Walsh, and Ramesh Johari. 2015. The New Stats Engine. *White Paper* (2015).

[15] Nornadiah Mohd Razali, Yap Bee Wah, et al. 2011. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics* 2, 1 (2011), 21–33.

[16] Fritz W Scholz and Michael A Stephens. 1987. K-sample Anderson–Darling tests. *J. Amer. Statist. Assoc.* 82, 399 (1987), 918–924.

[17] Richard Simard and Pierre L'Ecuyer. 2011. Computing the two-sided Kolmogorov-Smirnov distribution. *Journal of Statistical Software* 39 (2011), 1–18.

[18] A. Wald. 1945. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics* 16, 2 (1945), 117–186.

[19] Ya Xu, Weitao Duan, and Shaochen Huang. 2018. SQR: Balancing Speed, Quality and Risk in Online Experiments. *KDD* (2018).

[20] Ya Xu, Weitao Duan, and Shaochen Huang. 2018. SQR: Balancing speed, quality and risk in online experiments. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 895–904.

[21] Bilal Yurdakul and Joshua Naranjo. 2019. Statistical properties of the population stability index. *Journal of Risk Model Validation* 14, 4 (2019).

[22] Zezhong Zhang, Keyu Nie, and Ted Tao Yuan. 2020. Moving Metric Detection and Alerting System at eBay. *arXiv preprint arXiv:2004.02360* (2020).

[23] Zhenyu Zhao, Miao Chen, Don Matheson, and Maria Stone. 2016. Online experimentation diagnosis and troubleshooting beyond aa validation. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 498–507.

[24] Zhenyu Zhao, Mandie Liu, and Anirban Deb. 2018. Safely and quickly deploying new features with a staged rollout framework using sequential test and adaptive experimental design. In *2018 3rd International Conference on Computational Intelligence and Applications (ICCIA)*. IEEE, 59–70.

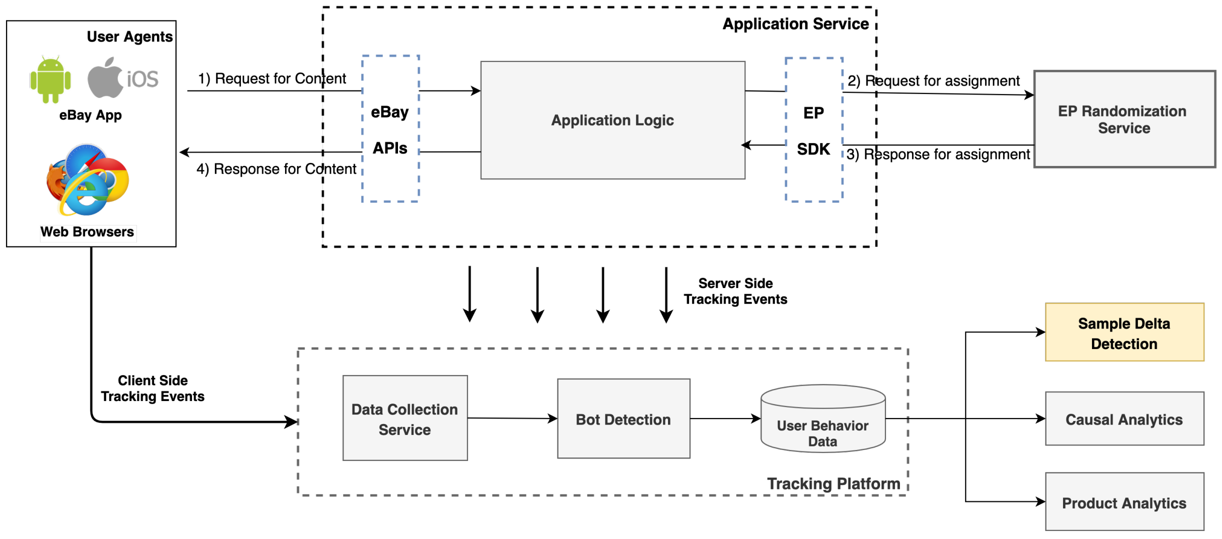

Figure 4: Experiment and Tracking Data Flow

A SEQUENTIAL PROBABILITY RATIO TEST

Consider a random variable Y with density function $f(y|p)$ parametrized by p. For the simple hypothesis:

$$H_0 : p = p_0$$
$$H_A : p \geq p_1$$

based on n independent observations $y_1, \ldots, y_n$. The log likelihood ratio is defined as:

$$\lambda_N = \log \frac{l_{H_A}}{l_{H_0}} = \log \frac{\prod_{j=1}^{n} f_1(y_j)}{\prod_{j=1}^{n} f_0(y_j)}$$

where $l_{H.}$ stands for the likelihood of observed samples based on distribution in $H.$, To simplify the notation, let $f_1 = f(y|p_1)$ and $f_0 = f(y|p_0)$. According to Neyman-Pearson theory ([8]), the most powerful test is given as the following:

$$\textbf{Reject } H_0 \textbf{ if } \lambda_n > r$$
$$\textbf{Accept } H_0 \textbf{ if } \lambda_n \leq r$$

The idea of sequential likelihood ratio testing is to add a third possibility in addition to accepting or rejecting, we can also elect to go on collecting data and decide later. Specifically, choose constants $A < B$ and sample $y_1, y_2, \ldots$ sequentially until the random stopping time

$$n_k = \min\{k : \lambda_k \notin [A, B]\}$$

at which point we reject H_0 if $\lambda_{n_k} > B$ and accept H_0 if $\lambda_{n_k} < A$. Wald [18] further showed that we can set $A = \log \frac{\beta}{1-\alpha}$ and $B = \log \frac{1-\beta}{\alpha}$ to control the overall type I error α and type II error β.

A.1 Gaussian case

In the Gaussian case, we assume $\hat{p}_k = \frac{x_k^T}{n_k}$ follows distribution $N(p_0, \sigma^2)$ under H_0, and $N(p_0 + \delta, \sigma^2)$ under H_A. The likelihood is given by

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(\hat{p}_k - p)^2\right]$$

Then the log likelihood ratio is:

$$\begin{aligned}
\lambda_k &= \log \frac{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(\hat{p}_k - p_0)^2\right]}{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(\hat{p}_k - p_0 - \delta)^2\right]} \\
&= -\frac{1}{2\sigma^2}(\hat{p}_K - p_0)^2 + \frac{1}{2\sigma^2}(\hat{p}_k - p_0 - \delta)^2 \\
&= -\frac{\delta^2 - 2\delta(\hat{p}_k - p_0)}{2\sigma^2}
\end{aligned}$$

Plug in the estimator $\hat{\sigma}_k^2 = \frac{\hat{p}_k(1-\hat{p}_k)}{n_k}$ for σ^2, we observe the formula in equation 7 : $t_k^A = -\frac{\delta^2 - 2\delta(\hat{p}_k - p_0)}{2\hat{\sigma}_k^2}$. Similarly we can also obtain t_k^B in equation 8 as well.

A.2 Binomial case

In the binomial case, we assume x_k^T following binomial distribution $B(n_k, p)$. The likelihood will be:

$$C_{x_k}^{n_k} p^{x_k} (1-p)^{n_k - x_k^T}$$

Suppose $p_0 = p_0$ and $p_1 = p_0 + \delta$. Then the log likelihood ratio is :

$$\begin{aligned}
\lambda_k &= \log \frac{C_{x_k^T}^{n_k} p_0^{x_k^T} (1-p_0)^{n_k - x_k^T}}{C_{x_k^T}^{n_k} (p_0 + \delta)^{x_k^T} (1 - p_0 - \delta)^{n_k - x_k^T}} \\
&= x_k \log(\frac{p_0 + \delta}{p_0}) + (n_k - x_k) \log(\frac{1 - p_0 - \delta}{1 - p_0})
\end{aligned}$$

So we obtain the formula in equation 9: $t_k^A = x_k^T log(\frac{p_0+\delta}{p_0}) + x_k^C log(\frac{1-p_0-\delta}{1-p_0})$. Similarly we could obtain equation 10 as well.

B SIMULATION ON DIFFERENT NOISE SCALE

We use same simulation settings in the Dataset 3, except we sample $x \sim Poisson(\lambda)$ with different $\lambda \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Hence, we can inject average noise from 0.05% to 0.15% to show the excellent performance of $\textbf{PSI}_{k=1}$ test under different noise scale.

	χ^2 test	AD test	KS test	$\textbf{PSI}_1$ test
0.05% noise	100%	23.0%	19.0%	73.0%
0.06% noise	100%	38.0%	31.0%	77.0%
0.07% noise	100%	44.0%	41.0%	93.0%
0.08% noise	100%	54.0%	50.0%	94.0%
0.09% noise	100%	78.0%	73.0%	100%
0.1% noise	100%	75.0%	75.0%	100%
0.11% noise	100%	75.0%	72.0%	100%
0.12% noise	100%	73.0%	75.0%	100%
0.13% noise	100%	82.0%	82.0%	100%
0.14% noise	100%	86.0%	89.0%	100%
0.15% noise	100%	93.0%	89.0%	100%

Table 7: Recall on different noise scale.

	χ^2 test	AD test	KS test	$\textbf{PSI}_1$ test
0.05% noise	63.69%	82.14%	86.36%	100%
0.06% noise	66.23%	86.36%	92.87%	100%
0.07% noise	68.02%	89.79%	94.61%	100%
0.08% noise	63.29%	93.73%	94.03%	100%
0.09% noise	65.36%	92.86%	94.81%	100%
0.10% noise	65.79%	96.15%	96.15%	100%
0.11% noise	64.52%	92.61%	96.00%	100%
0.12% noise	66.23%	94.81%	93.75%	100%
0.13% noise	63.69%	94.47%	98.79%	100%
0.14% noise	66.25%	96.63%	100%	100%
0.15% noise	63.50%	91.18%	97.81%	100%

Table 8: Precision on different noise scale.