21

Sample Ratio Mismatch and Other Trust-Related Guardrail Metrics

> The major difference between a thing that might go wrong and a thing that cannot possibly go wrong is that when a thing that cannot possibly go wrong goes wrong it usually turns out to be impossible to get at or repair
> – *Douglas Adams*

Why you care: *Guardrail metrics are critical metrics designed to alert experimenters about violated assumptions. There are two types of guardrail metrics: organizational and trust-related. Chapter 7 discusses organizational guardrails that are used to protect the business, and this chapter describes the Sample Ratio Mismatch (SRM) in detail, which is a trust-related guardrail. The SRM guardrail should be included for every experiment, as it is used to ensure the internal validity and trustworthiness of the experiment results. A few other trust-related guardrail metrics are also described here.*

As the Douglas Adams quotation shows, many people assume that the experiment will execute according to the design. When that assumption fails, and it fails more often than people expect, the analysis is usually heavily biased, and some conclusions are invalid. Multiple companies have reported seeing SRMs and have highlighted the value of this test as a guardrail (Kohavi and Longbotham 2017, Zhao et al. 2016, Chen, Liu and Xu 2019, Fabijan et al. 2019).

Sample Ratio Mismatch

The Sample Ratio Mismatch (SRM) metric looks at the ratio of users (or other units, see Chapter 14) between two variants, usually a Treatment and the Control. If the experiment design calls for exposing a certain ratio of users

(say 1:1) to the two variants, then the results should closely match the design. Unlike metrics that could be impacted by the Treatment, the decision to expose a user to a variant must be independent of the Treatment, so the ratio of the users in the variants should match the experimental design. For example, if you flip a fair coin 10 times and turn up 4 heads and 6 tails, a ratio of 0.67, it's no surprise. However, the Law of Large Numbers implies that, with high probability, the ratio should get closer to one as your sample size grows.

When the p-value for the Sample Ratio metric is low, that is, the probability of observing such a ratio or more extreme conditioned on the design ratio, there is a sample ratio mismatch (SRM), and all other metrics are probably invalid. You can use a standard t-test or chi-squared test to compute the p-value. An example Excel spreadsheet is available at http://bit.ly/srmCheck.

Scenario 1

In this experiment, Control and Treatment are each assigned 50% of users. You expect to see an approximately equal number of users in each, but your results are:

- Control: 821,588 users
- Treatment: 815,482 users

The ratio between the two is 0.993 whereas, per design, the ratio should be 1.0.

The p-value of the above .993 Sample Ratio is 1.8E-6, so the probability of seeing this ratio or more extreme, in a design with an equal number of users in Control and Treatment, is 1.8E-6, or less than 1 in 500,000!

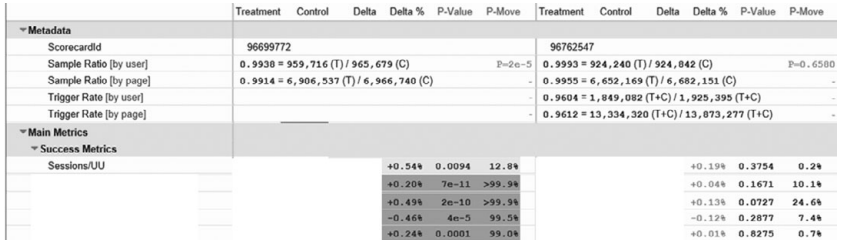
You just observed an extremely unlikely event. It is therefore more likely that there is a bug in the implementation of the experiment, and you should not trust any of the other metrics.

Scenario 2

This experiment also runs with Control and Treatment each assigned 50% of users and the ratio ends up at 0.994. You compute the p-value, and it's 2E-5, which is still very unlikely. It's a small percentage, so how far off can the metrics be? Do you really have to throw away the results?

Figure 21.1 shows an actual scorecard from Bing.

The middle column shows Treatment, Control, Delta, Delta %, P-Value, and P-Move (a Bayesian probability not germane for this example). The values for Treatment and Control are hidden to avoid disclosing confidential data, but they are not germane for our example. You can see that all five success metrics



	Treatment	Control	Delta	Delta %	P-Value	P-Move	Treatment	Control	Delta	Delta %	P-Value	P-Move
▾ Metadata												
ScorecardId	96699772						96762547					
Sample Ratio [by user]	0.9938 = 959,716 (T) / 965,679 (C)					P=2e-5	0.9993 = 924,240 (T) / 924,842 (C)					P=0.6580
Sample Ratio [by page]	0.9914 = 6,906,537 (T) / 6,966,740 (C)					-	0.9955 = 6,652,169 (T) / 6,682,151 (C)					-
Trigger Rate [by user]						-	0.9604 = 1,849,082 (T+C) / 1,925,395 (T+C)					-
Trigger Rate [by page]						-	0.9612 = 13,334,320 (T+C) / 13,873,277 (T+C)					-
▾ Main Metrics												
▾ Success Metrics												
Sessions/UU				+0.54%	0.0094	12.8%				+0.19%	0.3754	0.2%
				+0.20%	7e-11	>99.9%				+0.04%	0.1671	10.1%
				+0.49%	2e-10	>99.9%				+0.13%	0.0727	24.6%
				-0.46%	4e-5	99.5%				-0.12%	0.2877	7.4%
				+0.24%	0.0001	99.0%				+0.01%	0.8275	0.7%

Figure 21.1 Scorecard from Bing. The left column shows the meta-data, or metric names. The center column shows the statistics for each metric for the overall experiment. The right column shows the statistics for each metric for a segment of the population

improved, starting with Sessions/UU (UU = Unique User), and that the p-values are small (below 0.05 for all) to extremely small (below 0.0001 for the bottom four metrics).

The right column represents slightly over 96% of users; the excluded users were those that used an old version of the Chrome browser, which was the cause of the SRM. Also, a bot was not properly classified due to some changes in the Treatment, causing an SRM. Without the segment, the remaining 96% of users are properly balanced, and the metrics show no statistically significant movement in the five success metrics.

SRM Causes

There are many examples of SRMs causing incorrect results that have been documented (Zhao et al. 2016, Chen et al. 2019, Fabijan et al. 2019), and at Microsoft about 6% of experiments exhibited an SRM.

Here are some causes of SRMs:

- Buggy randomization of users. While simple Bernoulli randomization of users based on the percentages that get assigned to Control and Treatment is easy to imagine, things get more complex in practice because of ramp-up procedures discussed in Chapter 15 (e.g., starting an experiment at 1%, and ramping up to 50%), exclusions (users in experiment X should not be in experiment Y), and attempts to balance covariates by looking at historical data (see hash seed in Chapter 19).

  In one real example, the experiment Treatment was exposed to the Microsoft Office organization within Microsoft at 100%, and then an experiment was started exposing external users equally at 10%/10%. The

relatively small set of additional Office users in Treatment was enough to skew the results and make the Treatment look artificially good (as these are heavy users). The SRM provided a useful guardrail for trustworthiness of the result; when these internal-to-Microsoft users were removed, the strong Treatment effect disappeared.

- Data pipeline issues, such as the bot filtering mentioned in Scenario 2 above.
- Residual effects. Experiments are sometimes restarted after fixing a bug. When the experiment is visible to users, there is a desire not to re-randomize the users, so the analysis start date is set to the point after introducing the bug fix. If the bug was serious enough for users to abandon, then there will be an SRM (Kohavi et al. 2012).
- Bad trigger condition. The trigger condition should include any user that *could* have been impacted. A common example is a redirect: website A redirects a percentage of users to website A', their new website being built and tested. Because the redirect generates some loss, there will be typically be an SRM if only users that make it to website A'; are assumed to be in the Treatment. See Chapter 20.
- Triggering based on attributes impacted by the experiment. For example, suppose you are running a campaign on dormant users based on a *dormant* attribute stored in the user profile database. If the Treatment is effective enough to make some dormant users more active, then identifying users based on this attribute at the end of the experiment will give an SRM: The users that were dormant early and are now active will be excluded by the trigger condition. The analysis should be triggered to the state of the *dormant* attribute before the experiment started (or before each user was assigned). Trigger conditions based on machine learning algorithms are especially suspect because models may be updated while the experiment is running and impacted by Treatment effect.

Debugging SRMs

As noted above, when the p-value for the Sample Ratio guardrail metric is low, you should reject the hypothesis that the design is properly implemented and assume there is a bug someplace in the system. Don't even look at any other metrics, except to help debug what went wrong. Debugging an SRM is hard, and internal tools are usually built to help debug SRMs, such as by implementing some of the suggestions below.

Here are common directions to investigate that we have found to be useful:

- Validate that there is no difference upstream of the randomization point or trigger point. For example, if you are analyzing users starting at the checkout point because you changed a checkout feature, make sure that there is no difference between the variants upstream of that point. If you're evaluating 50% off vs. two-for-one at the checkout point, you cannot mention any of these options on the homepage; if you do, you must analyze users starting from the homepage.

  The Bing Image team runs experiments on users searching using Bing Image. They found that sometimes the experiment impacts the regular Bing web search results by serving image search results inline, often causing SRMs.

- Validate that variant assignment is correct. Are users properly randomized at the top of the data pipeline? While most variant-assignment systems start with simple randomization schemes based on hashing the user ID, assignment gets complicated over time supporting concurrent experiments and isolation groups, where different experiments are guaranteed to not be exposed to the same users (Kohavi et al. 2013).

  For example, suppose one experiment changes the font color from black to dark blue and a concurrent experiment starts that changes the background color, but that experiment filters to users who have their font set to black. Due to the way the code is run, the second experiment "steals" users from the first, but only from the font-set-to-black variant. Of course, this causes an SRM.

- Follow the stages of the data processing pipeline to see whether any cause an SRM. For example, a very common source of SRMs is bot filtering. Heuristics are typically used to remove bots, as they increase noise and make analysis less sensitive. At Bing over 50% of traffic in the United States is filtered out as bots, and 90% of traffic in China and Russia is bot generated! In one extreme case at MSN, the Treatment was so good at increasing usage, that the best users passed one heuristic threshold and were classified as bots – a bug. Except for triggering the SRM, the result appeared that the Treatment was significantly worse because the best users were excluded (Kohavi 2016).
- Exclude the initial period. Is it possible that both variants did not start together? In some systems, a Control is shared across multiple experiments. Starting the Treatment later can cause multiple problems even if the analysis period starts after the Treatment started. For example, caches take time to prime, apps take time to push, phones may be offline causing delays.

- Look at the Sample Ratio for segments.
  - Look at each day separately; was there an event on some day that was anomalous? For example, did someone ramp the experiment percentages for Treatment on some day? Or did another experiment start and "steal" traffic?
  - Is there a browser segment that stands out, as in Scenario 2 above?
  - Do new users and returning users show different ratios?
- Look at the intersection with other experiments. Treatment and Control should have similar percentages of variants from other experiments.

In some cases, if the SRM is understood, it is possible to fix the cause (e.g., bots) during the analysis phase. However, in other cases the removal of traffic (e.g., removal of browser due to a bug for that browser) implies that some segments have not been properly exposed to the Treatment, and it is better to rerun the experiment.

Other Trust-Related Guardrail Metrics

There are other metrics besides the SRM to indicate that something is wrong (Dmitriev et al. 2017). Sometimes these follow deep investigations and relate to software bugs, as the following examples show.

- Telemetry fidelity. Click tracking is typically done using web beacons, which is known to be lossy, that is, less than 100% of clicks are properly recorded (Kohavi, Messner et al. 2010). If the Treatment impacts the loss rate, the results may appear better or worse than the actual user experience. Having a metric to assess the loss, such as through internal referrers to the website or through clicks that use dual logging (sometimes used in ad clicks, which require high fidelity), fidelity issues may be uncovered.
- Cache hit rates. As noted in Chapter 3, shared resources can violate SUTVA (Kohavi and Longbotham 2010). Having metrics for shared resources, such as cache hit rates, may help identify unexpected factors that impact the trustworthiness of the experiment.
- Cookie write rate – the rate that the variant writes permanent (non-session) cookies. This phenomenon, dubbed *cookie clobbering* (Dmitriev et al. 2016), can cause severe distortion to other metrics due to browser bugs. One experiment at Bing wrote a cookie that was not used anywhere and set it to a random number with every search response page. The results showed massive user degradations in all key metrics, including sessions-per-user, queries-per-user, and revenue-per-user.

- Quick queries are two or more search queries that arrive from the same user to a search engine within one second of each other. Google and Bing have both observed this phenomenon, but to date have been unable to explain their cause. What we know is that some Treatments increase or decrease the proportion of quick queries and those results are deemed untrustworthy.