Online Experiments: Practical Lessons

Ron Kohavi, Roger Longbotham, and Toby Walker, *Microsoft*

When running online experiments, getting numbers is easy; getting numbers you can trust is hard.

From ancient times through the 19th century, physicians used bloodletting to treat acne, cancer, diabetes, jaundice, plague, and hundreds of other diseases and ailments (D. Wooton, *Doctors Doing Harm since Hippocrates*, Oxford Univ. Press, 2006). It was judged most effective to bleed patients while they were sitting upright or standing erect, and blood was often removed until the patient fainted. On 12 December 1799, 67-year-old President George Washington rode his horse in heavy snowfall to inspect his plantation at Mount Vernon. A day later, he was in respiratory distress and his doctors extracted nearly half of his blood over 10 hours, causing anemia and hypotension; he died that night.

Today, we know that bloodletting is unhelpful because in 1828 a Parisian doctor named Pierre Louis did a controlled experiment. He treated 78 people suffering from pneumonia with early and frequent bloodletting or less aggressive measures and found that bloodletting didn't help survival rates or recovery times.

Having roots in agriculture and medicine, controlled experiments have spread into the online world of websites and services. In an earlier Web Technologies article (R. Kohavi and R. Longobotham, "Online Experiments: Lessons Learned," *Computer,* Sept. 2007, pp. 85-87) and a related survey (R. Kohavi et al., "Controlled Experiments on the Web: Survey and Practical Guide," *Data Mining and Knowledge Discovery,* Feb. 2009, pp. 140-181), Microsoft's Experimentation Platform team introduced basic practices of good online experimentation.

Three years later and having run hundreds of experiments on more than 20 websites, including some of the world's largest, like msn.com and bing.com, we have learned some important practical lessons about the limitations of standard statistical formulas and about data traps. These lessons, even for seemingly simple univariate experiments, aren't taught in Statistics 101. After reading this article, we hope you'll have better negative introspection: to know what you don't know.

ONLINE EXPERIMENTS: A QUICK REVIEW

In an online controlled experiment, users are randomly assigned to two or more groups for some period of time and exposed to different variants of the website. The most common online experiment, the A/B test, has two variants: the A version of the site is the *control* and the B version is the *treatment*.

The experimenters define an *overall evaluation criterion* (OEC) and compute a statistic—for example, the mean of the OEC—for each variant. The OEC statistic is also referred to as a *key performance indicator* (KPI); in statistics, the OEC is often called the response or dependent variable.

The difference between the OEC statistic for the treatment and control groups is the *treatment effect*. If the experiment was designed and executed properly, the only thing consistently different between the two variants is the planned change between the control and treatment. Consequently, any statistically significant effect is the result of the planned change, establishing causality with high probability.

Common extensions to simple A/B tests include multiple variants along a single axis—for example, A/B/C/D—and multivariable tests that expose users to changes along several axes, such as font choice, size, and color.

LIMITATIONS OF STATISTICAL FORMULAS

Most online experiments randomly assign users to variants based on a user ID stored in a cookie. Thus "users" are statistically independent of one another, and any metric calculated by user can employ standard statistical methods requiring independence—t-tests, analysis of variance (ANOVA), and so on. Standard metrics like clicks per user work well for this analysis, but many important metrics aren't defined by user but by, for example, page view or session.

One metric used extensively in online experiments is *click-through rate* (CTR), which is the total number of clicks divided by the total number of impressions. This formula can be used to estimate the average CTR, but estimating the standard deviation is difficult. It's tempting to treat this as a sequence of independent Bernoulli trials, with each page view either getting a click or not, but this will underestimate the standard deviation, and too many experiments will incorrectly appear as statistically significant. The assumption of independence of page views doesn't hold because clicks and page views from the same user are correlated. For the same reason, metrics calculated by user-day or by session are also correlated.

To address this problem, we use both bootstrapping and the delta method. Bootstrapping "resamples" the dataset to estimate the standard deviation. It's computationally intensive, especially for large datasets. The delta method, on the other hand, is computationally simpler because it uses a simple, low-order Taylor series approximation to compute the variance. For large sample sizes it works well because higher-order terms converge to zero.

AVOIDABLE DATA TRAPS

In our work we've encountered many common data traps that are avoidable. Some are seemingly obvious; others are obvious in hindsight, but the hindsight was painfully won.

Variant assignment

A basic requirement of variant assignment is that randomization work correctly. We've observed subtle examples of variant assignment bias that significantly impacted the results. For example, in one Bing experiment, a small misconfiguration caused internal Microsoft traffic to always be assigned to the control. In another experiment on the MSN homepage, cobranded users, whose page is slightly different, were always shown the control. Even a small 1 percent imbalance between the control and treatment can easily cause a 5 percent delta to the treatment effect. Such users, if not assigned randomly, must be excluded from the analysis.

Using browser redirects to send users to a variant if they aren't in the control introduces subtle biases,including performance differences.

User identification

User IDs are typically stored in browser cookies. In Microsoft's ecosystem, which has multiple domains (msn.com, microsoft.com, bing.com, live.com), synchronization of identity across domains is complex. Because user assignment to variants is based on user ID, the user's experience may "flip" to another variant if a user's cookie changes due to the identity synchronization. In addition, if one of the variants updates the cookie more frequently, there will be a consistent bias.

Cookie-related bugs are extremely hard to debug. In one experiment, the Microsoft support site cached page outputs and didn't update the cookies properly only for one variant, causing a significant bias. It's therefore important to check for consistent allocation of the users against the target percentages.

Data loss

Online experiments are vulnerable to data loss at all stages of the pipeline: logging, extract, transform, and load. We recommend monitoring for data loss at both the aggregate and machine level. At best, the data loss is unbiased and you'll have lost some sample power, but in some cases, the loss can bias the results. In one experiment, users were sent to one website in the control and a different one in the treatment. The website in the control redirected non-US users to their international site, which wasn't logging to the same system, and they were "lost" to the experiment. Using browser redirects to send users to a variant if they aren't in the control also introduces subtle biases, including performance differences.

Event tracking

Most online experiments log clicks to the servers using some form of callback, such as JavaScript handlers or URL redirects. Such tracking mechanisms can cause problems. In one experiment, an unexpectedly strong negative impact on CTR surfaced. An investigation revealed a problem with the callback mechanism used for tracking clicks that affected only Internet Explorer 6 (IE6). Another experiment used redirects for tracking ads in the control and JavaScript callbacks for the treatment. Because the loss rate for these is different and because some robots don't respect JavaScript, there was significant bias in click-through metrics.

Server differences

A tempting experimental design is to run one variant—say, the control—

on an existing server fleet and the treatment on another fleet. If there are systematic differences in the servers—different hardware capabilities, a different network, patches, more crashes, different data centers, and so on—any or all of these can severely bias the experiment.

System-wide interference

Because online experiments for the same site typically run in the same physical systems, experiment variants can interfere with one another or with other experiments when running at the same time. For example, if variants use a shared resource—say, a least recently used (LRU) cache—differential use of that resource can have unexpected consequences on the entire system and thus the experiments.

For example, in one experiment, many key performance metrics went down when a small change was made to how results were generated for a page. The cause turned out to be an unexpected cache-hit-rate difference between the control and treatment. Websites cache results for commonly requested pages or searches to improve performance. While both the control and treatment cached results in the same cache, they used different keys for the same request. Because the control received much more traffic than the treatment, it occupied a much larger part of the shared cache. The result was that the treatment had a much lower cache hit rate than the control, resulting in significantly slower performance.

The solution in this case was simple: create a special "dummy" control whose allocation matches the treatment; this equalizes the cache hit rate of the control and treatment. The lesson: look for timing differences in the user experience between variants.

UNAVOIDABLE DATA TRAPS

It's harder to keep other data traps from affecting your online experiment.

Robots

Robot requests are website requests that don't come from users' direct interaction with the website. Robot traffic can have nonmalicious sources—ranging from browser addins that prefetch pages linked from the current one, to search-engine crawlers, to a graduate student scraping webpages—as well as malicious sources like ad-click spam. For a large search engine or online portal site, robots can commonly represent 15 to 30 percent of page views.

Robots can easily bias experimental results because some robots have many more events—page views, clicks, searches, and so on—than members of the true user population.

It's naïve to hope that robots won't affect experimental results. Almost all experiments are conducted to optimize the website for human users. Not only are robots not part of the population of interest, adding background noise, they can seriously impact the conclusions.

Robots can easily bias experimental results because some robots have many more events—page views, clicks, searches, and so on—than members of the true user population. This biases the mean and increases the variance of the groups into which they're randomized. Increased variance means less chance to detect effects. For example, a robot we discovered in one experiment claimed it was IE8 yet "clicked" 600,000 times in a month. A single robot like this significantly skews multiple metrics if not detected and removed.

Some metrics are more sensitive to robot bias than others. Any metric with no upper bound is much more sensitive to outliers than metrics having a bounded value. For example, a metric that computes average number of queries per user can be heavily influenced by a single robot that issues thousands of queries.

Metrics such as conversion rate that take only 0 or 1 for each user are least affected by robots or outliers. When business requirements dictate nonrobust metrics, we recommend calculating related robust metrics and investigating directional differences—for example, from robots.

Detecting and removing robots is both critical and challenging. Most robots, especially nonmalicious ones, can be removed using simple techniques. For example, you could remove users with an unlikely large number of queries, who issue a significant number of queries but never click, or who issue queries too rapidly; you could also blacklist known user agents. Some robots will still slip through, so it's a good practice to use manual or automated methods to look for outliers. Finally, you should use robust statistical methods—nonparametric or rank-based techniques, robust ANOVA, and so on—to determine whether the data are similar to those of a standard analysis using, say, t-tests or ANOVA. A large discrepancy in the results warrants further investigation.

Finally, note that robot detection is an adversarial AI-complete problem: your goal is to identify the nonhuman robots, while adversaries write robots that masquerade as humans—for example, using distributed botnets. Above all, be vigilant.

Configuration drift

In a typical online system, operation is controlled by many different configuration settings distributed across multiple systems. These settings change while the experiment runs. There's a danger of the control and treatment drifting apart so they no longer match the original experiment's design. A good example is a bug fix made only to the control.

IDENTIFYING DATA TRAPS USING STATISTICS

We use a three-step process to help validate experimental data:

- logging audits,
- A/A experiments, and
- data validation checks.

We've previously written about the importance of conducting logging audits and A/A experiments before doing the first A/B experiment ("Controlled Experiments on the Web: Survey and Practical Guide"). As an experiment is running, you should conduct ongoing and final data validation checks. While there are nonstatistical checks that you should always do—for example, check for time periods with missing data—here we focus on statistical data validation checks.

We recommend defining *statistical integrity constraints*, criteria that should hold with high probability even if there's a large treatment effect, and checking them with statistical hypothesis tests. For example, if an experiment assigns 50 percent of users to the control and 50 percent to the treatment, you could use a binomial or chi-squared test to check that the percent in each isn't significantly different from the expected 50 percent. If the percentages turned out to be, say, 50.1 and 49.9, that could mean almost a 0.5 percent bias in key metrics depending on the reason for the imbalance. If the p-value of the test is small, you should assume some bias could exist and investigate.

Data-quality metrics are an important type of statistical integrity constraint. These are metrics for both the control and treatment that the experiment shouldn't affect—for example, cache hit rates. A significant difference in a data-quality metric calls for further investigation.

We also recommend looking for unexpected changes over time in an experiment's key statistics: samples sizes, treatment effect, means, and variance. Our analysis of one experiment showed a 2 percent increase of the treatment over the control. However, a graph of the treatment and control means over time revealed an unexpected increase in the treatment effect for a seven-hour period, as Figure 1 shows. We determined that the increase was due to an uncontrolled difference in the websites during this period (editorial error); after we removed the data for this seven-hour period, there was no longer a difference between the treatment and control.

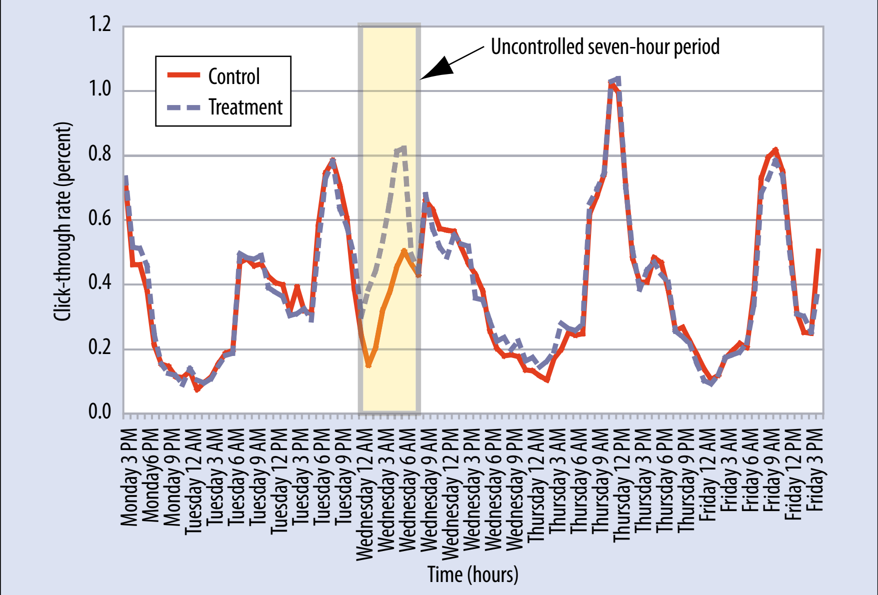

Figure 1. Hourly click-through rate (CTR) for the treatment and control of one experiment revealed an uncontrolled difference in the websites during a seven-hour period.

Because data issues are the norm, not the exception, in online experiments, we advocate a healthy degree of "data paranoia." Finding and understanding such issues are key to getting the right results but also require a substantial investment of time and energy. Even the smallest anomalies can lead to new insights. Getting numbers is easy; getting numbers you can trust is hard.

__Ron Kohavi__ is the general manager of Microsoft's Experimentation Platform, __Roger Longbotham__ is the team's analytics manager, and __Toby Walker__ works on the Bing Data Mining Team as technical lead for experiment analytics. Contact them at http://exp-platform.com.

Editor: Simon S.Y. Shim, Dept. of Computer Engineering, San Jose State Univ., San Jose, CA; simon.shim@sjsu.edu