Online Experiments: Lessons Learned

Ron Kohavi and Roger Longbotham
Microsoft

Web experiments generate insights and promote innovation.

While at Amazon.com from 1997 to 2002, Greg Linden created a prototype system that made personalized recommendations to customers when they placed items in their shopping cart (http://glinden.blogspot.com/2006/04/early-amazon-shopping-cart.html). The prototype looked promising, but "a marketing senior vice-president was dead set against it," claiming it would distract people from checking out, and Linden was forbidden to work on it further.

Nonetheless, Linden ran an online controlled experiment, and the "feature won by such a wide margin that not having it live was costing Amazon a noticeable chunk of change. With new urgency, shopping cart recommendations launched." Since then, multiple commercial Web sites have imitated Amazon.com's feature.

The Web provides an unprecedented opportunity to evaluate proposed changes or new features quickly using controlled experiments. The simplest experiments randomly assign live users to one of two variants: the *control*, which is commonly the existing version, and the *treatment*, which is usually a new version being evaluated.

Experimenters first collect metrics of interest, from runtime performance to implicit and explicit user behaviors and survey data. They then conduct statistical tests on this data to determine whether a significant difference exists between the two variants, which in turn leads them to retain or reject the null hypothesis that there's no difference between the versions.

Online experiments offer substantial benefits, but many pitfalls can trip up practitioners. Our work at Microsoft and other companies including Amazon.com, Blue Martini Software, Capital One, and Xerox has led to some important lessons in properly conducting such experiments. A practical guide is available at http://exp-platform.com/hippo.aspx.

OVERALL EVALUATION CRITERION

A common pitfall in Web experiments is the use of multiple metrics. For an organization that seeks to run many experiments in a given domain, it's strongly desirable to select a single quantitative measure, or *overall evaluation criterion* (OEC), to help determine whether a particular treatment is successful or not.

Consider an experiment with 25 different metrics in which three organizational teams have a stake in the outcome. How do the teams decide whether to launch the treatment if some of the metrics are positive and some are negative? Worse, if the metrics are favorable for one team and negative for another, deciding what to do could be contentious.

Also, with 25 metrics, even if the experiment has no effect, we should expect one or more metrics to appear statistically significantly different when using the common 95 percent confidence intervals. Having a single metric for decision making simplifies the process, clarifies the interpretations, and aligns the organization behind a clear objective.

The OEC can be a simple metric that summarizes important business goals or a weighted combination of metrics, as is often used in credit scores. It should reflect long-term objectives such as higher revenue, more users, or greater user engagement. In many cases, the OEC is an estimate of users' lifetime value.

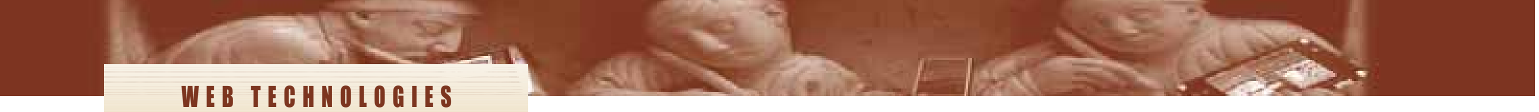
Factors that might positively impact the OEC include higher visit frequency, longer sessions, and disclosure of personal information such as zip codes and hobbies that can be used to improve the user experience. For retail sites, adding to a cart or wish list increases the probability of a future purchase, even if the transaction didn't occur in the current session. Users who have a large social network, sometimes called "connectors" or "sneezers," can have a much larger influence and thus a higher value in the OEC.

RANDOMIZATION

Good experimental design calls for blocking or randomizing over *nuisance factors* that impact the OEC but aren't of interest, such as the time of day, the day of the week, and the server that handles the request. Because time is a critical factor, running the control and treatment concurrently is essential for online experiments and far superior to interrupted time series.

Server fleets

Using different server fleets for the control and treatment can skew experimental results. Suppose, for example, server fleet F1 runs the control, and a newer server fleet F2, which handles



requests faster, runs the new treatment. This discrepancy introduces an unintentional bias. One way to identify such biases and other problems with user assignment is to run an A/A, or null, test in which the control and treatment are the same.

Representative environment

The experimental environment should represent the eventual environment as closely as possible. For example, if a Web site runs an experiment for three weekdays but has a different set of users during weekends, the results might not generalize. The experiment should run for multiple weeks to determine whether significant differences exist between weekdays and weekends. Partial days shouldn't be included for the same reason: Morning users might react differently to the treatment than afternoon, evening, or overnight users. Likewise, running an experiment during the Christmas season might not generalize to the rest of the year.

Hashing function

Randomization is too important to be left to chance. A common way to maintain user experience consistency is to employ a hashing function on a user ID stored in a cookie. Picking a good hashing function is critical—some aren't random enough. Cryptographic hashes such as MD5 are generally the best.

Failure to randomize properly can confound results when running multiple tests simultaneously. For example, computing separate hashes of both the experiment name and user ID, and then executing the final XOR at assignment time, produces severe correlations between experiments.

Assuming two experiments with two variants each running at 50/50, if the most significant bit of the hashes of the experiment names matched, users would always get the same assignment across both experiments; if they didn't match, users would get exactly the opposite assignment. Automated checks can ensure that user assignment is random and matches the design proportions.

Opt in/opt out

Letting users opt in or out of an experiment invalidates the randomness. Opt in especially is usually a bad idea. Clickthroughs for users that opt in are significantly higher, but these often don't materialize when all users switch to the new format. One reason for this is that people who opt in are likely to be heavy users whose clickthrough rate is already higher.

A significant portion of traffic to most Web sites consists of one-click sessions—users who enter the site and never click again. Some of these users are nonhuman bots and crawlers, and some are people who go to the site by mistake. Whatever the reason, this "bad" traffic never opts in, thus opt-in treatments have a selection bias.

Failure to randomize properly can confound results when running multiple tests simultaneously.

MINIMUM DURATION

An important but often overlooked step in a controlled experiment is planning for sufficient sample size, that is, statistical power. For most online experiments, this translates to how long to run the experiment.

Power calculation

It's not uncommon to run a test on a small population—say, one percent of users—and discover four weeks later that the test must continue for 10 more weeks to detect the expected size change. Power calculations help plan how long an experiment should run and what percentage of the population to expose.

To accomplish this, first determine the *sensitivity*, or how large a change in the OEC you want to be able to detect. For example, if your OEC is revenue per user and the mean is \$7 on your site, you might want a sensitivity of 1 percent, or the ability to detect a change to the mean of \$0.07. Then, estimate your OEC's standard deviation from historical data or an A/A test.

Assuming that you will do *t*-tests against the control, apply the formula $n = (16 \times \sigma^2)/\Delta^2$, where n is the number of users in each variant (assumed to be of equal size), σ^2 is your OEC's variance, and Δ is the sensitivity. The coefficient of 16 provides 80 percent power—the sample size provided by the formula gives an 80 percent probability of rejecting the null hypothesis that there's no difference between the treatment and control if the true mean differs from the true control by Δ. Replacing the 16 with 21 will increase the power to 90 percent.

Once you have your sample size, calculate how long you'll need to run the test based on normal traffic to the Web site and preferably round up to whole weeks.

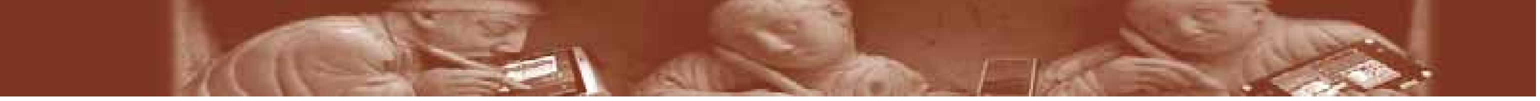
Overlapping experiments

Novice experimenters overly concerned about interactions tend to run experiments sequentially or on small disjointed subsets of users. However, unless there's a good reason to believe that experiments interact strongly, second-order effects are usually weaker than main effects.

Determining whether significant interactions occurred post hoc through, say, pairwise testing, is relatively easy. It's more important to run experiments on a large percentage of users to have sufficient power to detect small effects. Four independently randomized experiments that are concurrent and overlapping, each splitting users equally into control and treatment groups, are generally preferable to splitting the population into a 20 percent control group and four 20 percent treatment groups.

Concurrent univariate experiments

Commercial marketing literature suggests that univariate experiments that test one factor at a time are inferior to multivariate experiments, which vary many variables simultaneously according to a specialized



design such as fractional factorials. However, this is akin to substituting polynomial for linear models: It sounds good in theory, but the complexity leads to linear models being used more often in practice.

Running multiple univariate experiments concurrently offers several advantages: Univariate analysis is much easier for end users to understand, experiments can be turned on and off when bugs are found without shutting down other experiments, and any interactions among the factors can still be estimated as if using a full-factorial design.

ANALYSIS

Because they contain new code for the treatments, online experiments have a higher failure rate and more bugs. On the other hand, unlike offline experiments, analysis of online experiments should begin as soon as possible, that is, in near real time.

Bugs

Suppose you're looking for a 1 percent improvement in the treatment, but it has some bugs and key metrics are degrading by 10 percent. Because the power formula is quadratic in the effect size (Δ), you can detect a 10 percent degradation 100 times faster than a 1 percent change—thus, an experiment planned to run for two weeks to detect a 1 percent delta will have enough power to detect a 10 percent drop in less than 4 hours.

Consider defining bounding boxes for some metrics. For example, if the time to generate a page increases above a preset threshold too often, it's likely there's a bug, and you should abort the experiment.

Ramping up the percentage assigned to treatment over time is also recommended. Start with a small percentage, check that the metrics are within reasonable bounds, and only then increase the percentage assigned to the treatment. Ramping up over a day or two makes it possible to catch egregious errors while the experiment is exposed to a small population. More sophisticated analysis must be done if you combine time periods in which the treatment percentage changes.

Primacy and newness effects

Users accustomed to a particular feature might reject a new one, even if it's better. Conversely, a flashy new widget might initially attract users. In these cases, it's important to run the experiment longer and evaluate users' behavior after multiple exposures to the feature. Analysis of trends in the OEC is also helpful.

Secondary metrics

Experimenters often ignore secondary metrics that impact the user experience such as JavaScript errors, customer-service calls, and Web-page loading time. Experiments at Amazon.com showed that every 100-ms increase in the page load time decreased sales by 1 percent, while similar work at Google revealed that a 500-ms increase in the search-results display time reduced revenue by 20 percent.

Analysis of online experiments should begin as soon as possible, that is, in near real time.

Data cleansing

Bots and crawlers can account for 5 to 40 percent of a Web site's page views. Because many robots are from search engines, you want them to crawl the site. In most cases, robots don't send cookies, and they tend to distribute across the control and treatments, simply adding noise. However, some do send cookies and can significantly impact statistics and the OEC.

Looking for outliers and investigating them is important to detect and remove possible instances of fraud. For example, sellers on online retail sites might buy their own products to make a top-sellers list. The orders, usually of large quantities, are either cancelled later, returned, or, in cases where the seller is also the shipper, not actually shipped.

Finally, in our experience, data from Web sites is fragile, especially that from the online logs. Always check for metrics that get contaminated in some way. Automatic data validation checks are very helpful.

Falling for features

Two other common mistakes in online experiments are launching a feature that is statistically significantly different but has little business value and launching a feature because it doesn't negatively impact users.

Consider, for example, a feature that improves on some key metric and the difference is statistically significant, but the delta is 0.01 percent. The business impact is negligible, but the cost of maintaining the feature is high, outweighing any benefit of the feature.

As another example, suppose the CEO's favorite new feature isn't significantly different but there's a strong push to deploy it because the project development team worked hard. Not only are there maintenance costs involved, but the experiment could be underpowered and might not detect that the feature has a lower OEC.

Online experiments, whether they fail or succeed, generate insights that can bring a quick return on investment and promote innovation. We reserve the most important lesson for the end, and it's called Twyman's law: Any statistic that appears interesting is almost certainly a mistake. Make sure to double-check your data and computations. ■

Ron Kohavi is the general manager of Microsoft's Experimentation Platform, and Roger Longbotham is the team's lead statistician. Contact them at http://exp-platform.com.

Editor: Simon S.Y. Shim, Director of Development, SAP Labs, Palo Alto, CA; simon.shim@sap.com.