Data Min Knowl Disc (2009) 18:140–181
DOI 10.1007/s10618-008-0114-1

Controlled experiments on the web: survey and practical guide

Ron Kohavi · Roger Longbotham · Dan Sommerfield · Randal M. Henne

Received: 14 February 2008 / Accepted: 30 June 2008 / Published online: 30 July 2008

Abstract The web provides an unprecedented opportunity to evaluate ideas quickly using controlled experiments, also called randomized experiments, A/B tests (and their generalizations), split tests, Control/Treatment tests, MultiVariable Tests (MVT) and parallel flights. Controlled experiments embody the best scientific design for establishing a causal relationship between changes and their influence on user-observable behavior. We provide a practical guide to conducting online experiments, where end-users can help guide the development of features. Our experience indicates that significant learning and return-on-investment (ROI) are seen when development teams listen to their customers, not to the Highest Paid Person's Opinion (HiPPO). We provide several examples of controlled experiments with surprising results. We review the important ingredients of running controlled experiments, and discuss their limitations (both technical and organizational). We focus on several areas that are critical to experimentation, including statistical power, sample size, and techniques for variance reduction. We describe common architectures for experimentation systems and analyze their advantages and disadvantages. We evaluate randomization and hashing techniques, which we show are not as simple in practice as is often assumed. Controlled

Responsible editor: R. Bayardo.

R. Kohavi (✉) · R. Longbotham · D. Sommerfield · R. M. Henne
Microsoft, One Microsoft Way, Redmond, WA 98052, USA
e-mail: ronnyk@microsoft.com

R. Longbotham
e-mail: rogerlon@microsoft.com

D. Sommerfield
e-mail: dans@microsoft.com

R. M. Henne
e-mail: rhenne@microsoft.com

experiments typically generate large amounts of data, which can be analyzed using data mining techniques to gain deeper understanding of the factors influencing the outcome of interest, leading to new hypotheses and creating a virtuous cycle of improvements. Organizations that embrace controlled experiments with clear evaluation criteria can evolve their systems with automated optimizations and real-time analyses. Based on our extensive practical experience with multiple systems and organizations, we share key lessons that will help practitioners in running trustworthy controlled experiments.

Keywords Controlled experiments · A/B testing · e-commerce · Website optimization · MultiVariable Testing · MVT

1 Introduction

> *One accurate measurement is worth more than a thousand expert opinions*
> – Admiral Grace Hopper

In the 1700s, a British ship's captain observed the lack of scurvy among sailors serving on the naval ships of Mediterranean countries, where citrus fruit was part of their rations. He then gave half his crew limes (the Treatment group) while the other half (the Control group) continued with their regular diet. Despite much grumbling among the crew in the Treatment group, the experiment was a success, showing that consuming limes prevented scurvy. While the captain did not realize that scurvy is a consequence of vitamin C deficiency, and that limes are rich in vitamin C, the intervention worked. British sailors eventually were compelled to consume citrus fruit regularly, a practice that gave rise to the still-popular label *limeys* (Rossi et al. 2003; Marks 2000).

Some 300 years later, Greg Linden at Amazon created a prototype to show personalized recommendations based on items in the shopping cart (Linden 2006a, b). You add an item, recommendations show up; add another item, different recommendations show up. Linden notes that while the prototype looked promising, "a marketing senior vice-president was dead set against it," claiming it will distract people from checking out. Greg was "forbidden to work on this any further." Nonetheless, Greg ran a 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 sites have copied cart recommendations.

The authors of this paper were involved in many experiments at Amazon, Microsoft, Dupont, and NASA. The culture of experimentation at Amazon, where data trumps intuition (Kohavi et al. 2004), and a system that made running experiments easy, allowed Amazon to innovate quickly and effectively. At Microsoft, there are multiple systems for running controlled experiments. We describe several architectures in this paper with their advantages and disadvantages. A unifying theme is that controlled experiments have great return-on-investment (ROI) and that building the appropriate infrastructure can accelerate innovation. Stefan Thomke's book title is well suited here: *Experimentation Matters* (Thomke 2003).

The web provides an unprecedented opportunity to evaluate ideas quickly using controlled experiments, also called randomized experiments (single-factor or factorial designs), A/B tests (and their generalizations), split tests, Control/Treatment, and parallel flights. In the simplest manifestation of such experiments, live users are randomly assigned to one of two variants: (i) the Control, which is commonly the "existing" version, and (ii) the Treatment, which is usually a new version being evaluated. Metrics of interest, ranging from runtime performance to implicit and explicit user behaviors and survey data, are collected. Statistical tests are then conducted on the collected data to evaluate whether there is a statistically significant difference between the two variants on metrics of interest, thus permitting us to retain or reject the (null) hypothesis that there is no difference between the versions. In many cases, drilling down to segments of users using manual (e.g., OLAP) or machine learning and data mining techniques, allows us to understand which subpopulations show significant differences, thus helping improve our understanding and progress forward with an idea.

Controlled experiments provide a methodology to reliably evaluate ideas. Unlike other methodologies, such as post-hoc analysis or interrupted time series (quasi experimentation) (Charles and Melvin 2004), this experimental design methodology tests for causal relationships (Keppel et al. 1992, pp. 5–6). Most organizations have many ideas, but the return-on-investment (ROI) for many may be unclear and the evaluation itself may be expensive. As shown in the next section, even minor changes can make a big difference, and often in unexpected ways. A live experiment goes a long way in providing guidance as to the value of the idea. Our contributions include the following.

- In Sect. 3 we review controlled experiments in a web environment and provide a rich set of references, including an important review of statistical power and sample size, which are often missing in primers. We then look at techniques for reducing variance that we found useful in practice. We also discuss extensions and limitations so that practitioners can avoid pitfalls.
- In Sect. 4, we present several alternatives to MultiVariable Tests (MVTs) in an online setting. In the software world, there are sometimes good reasons to prefer concurrent uni-variate tests over traditional MVTs.
- In Sect. 5, we present generalized architectures that unify multiple experimentation systems we have seen, and we discuss their pros and cons. We show that some randomization and hashing schemes fail conditional independence tests required for statistical validity.
- In Sect. 6 we provide important practical lessons.

When a company builds a system for experimentation, the cost of testing and experimental failure becomes small, thus encouraging innovation through experimentation. Failing fast and knowing that an idea is not as great as was previously thought helps provide necessary course adjustments so that other more successful ideas can be proposed and implemented.

2 Motivating examples

The fewer the facts, the stronger the opinion
– Arnold Glasow

The following examples present surprising results in multiple areas. The first two deal with small UI changes that result in dramatic differences. The third example shows how controlled experiments can be used to make a tradeoff between short-term revenue from ads and the degradation in the user experience. The fourth example shows the use of controlled experiments in backend algorithms, in this case search at Amazon.

2.1 Checkout page at Doctor FootCare

The *conversion rate* of an e-commerce site is the percentage of visits to the website that include a purchase. The following example comes from Bryan Eisenberg's articles (Eisenberg 2003a, b).

Can you guess which one has a higher conversion rate and whether the difference is significant?

There are nine differences between the two variants of the Doctor FootCare checkout page shown in Fig. 1. If a designer showed you these and asked which one should be deployed, could you tell which one results in a higher conversion rate? Could you estimate what the difference is between the conversion rates and whether that difference is significant?

We encourage you, the reader, to think about this experiment before reading the answer. Can you estimate which variant is better and by how much? It is very humbling to see how hard it is to correctly predict the answer.

Please, challenge yourself!

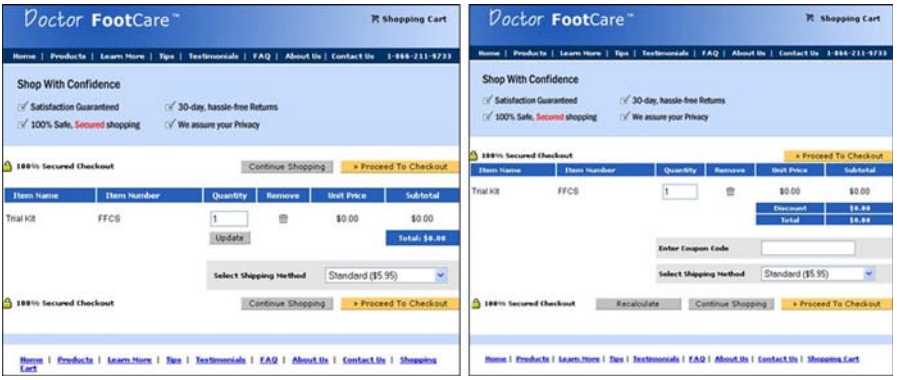


Fig. 1 Variant A on left, Variant B on right

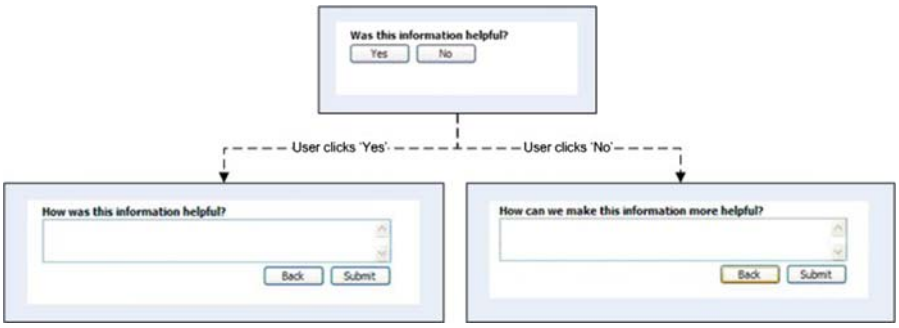


Fig. 2 Microsoft help ratings widget. The original widget is shown above. When users click on Yes/No, the dialogue continues asking for free-text input (two-phase)

Variant A in Fig. 1 outperformed variant B by an order of magnitude. In reality, the site "upgraded" from the A to B and lost 90% of their revenue! Most of the changes in the upgrade were positive, but the coupon code was the critical one: people started to think twice about whether they were paying too much because there are discount coupons out there that they do not have. By removing the discount code from the new version (B), conversion-rate increased 6.5% relative to the old version (A) in Fig. 2.

2.2 Ratings of Microsoft Office help articles

Users of Microsoft Office who request help (or go through the Office Online website at http://office.microsoft.com) are given an opportunity to rate the articles they read. The initial implementation presented users with a *Yes/No* widget. The team then modified the widget and offered a 5-star ratings.

The motivations for the change were the following:

1. The 5-star widget provides finer-grained feedback, which might help better evaluate content writers.
2. The 5-star widget improves usability by exposing users to a single feedback box as opposed to two separate pop-ups (one for *Yes/No* and another for *Why*).

Can you estimate which widget had a higher response rate, where response is any interaction with the widget?

The surprise here was that number of ratings plummeted by about 90%, thus significantly missing on goal #2 above. Based on additional tests, it turned out that the two-stage model helps in increasing the response rate. Specifically, a controlled experiment showed that the widget shown in Fig. 3, which was a two-stage model and also clarified the 5-stars direction as "Not helpful" to "Very helpful" outperformed the one in Fig. 4 by a factor of 2.2, i.e., the response rate was 2.2 times higher.

Even goal #1 was somewhat of a disappointment as most people chose the extremes (one or five stars). When faced with a problem for which you need help, the article either helps you solve the problem or it does not!

The team finally settled on a yes/no/I-don't-know option, which had a slightly lower response rate than just yes/no, but the additional information was considered useful.

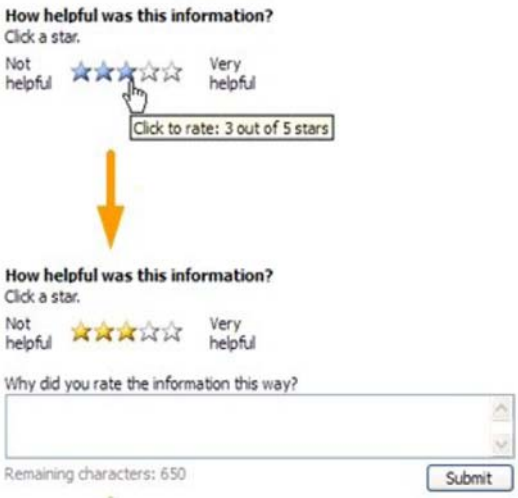


Fig. 3 A two-stage model widget

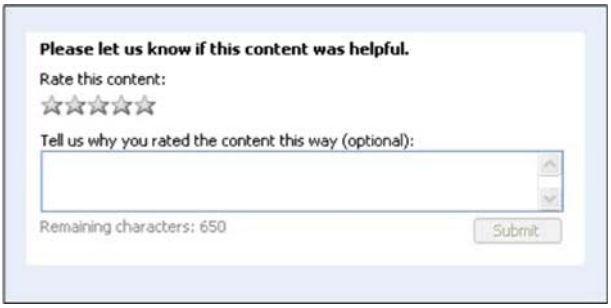


Fig. 4 New 5-star ratings widget. Single rating widget with 5 stars

2.3 MSN home page ads

A critical question that many site owners face is how many ads to place. In the short-term, increasing the real-estate given to ads can increase revenue, but what will it do to the user experience, especially if these are non-targeted ads? The tradeoff between increased revenue and the degradation of the end-user experience is a tough one to assess, and that's exactly the question that the MSN home page team at Microsoft faced in late 2007.

The MSN home page is built out of modules. The Shopping module is shown on the right side of the page above the fold. The proposal was to add three offers right below it, as shown in Fig. 5, which meant that these offers would show up below the fold for most users. The Display Ads marketing team estimated they could generate tens of thousands of dollars per day from these additional offers.

The interesting challenge here is how to compare the ad revenue with the "user experience." In Sect. 3.1, we refer to this problem as the OEC, or the Overall Evaluation Criterion. In this case, we decided to see if page views and clicks decreased, and assign a

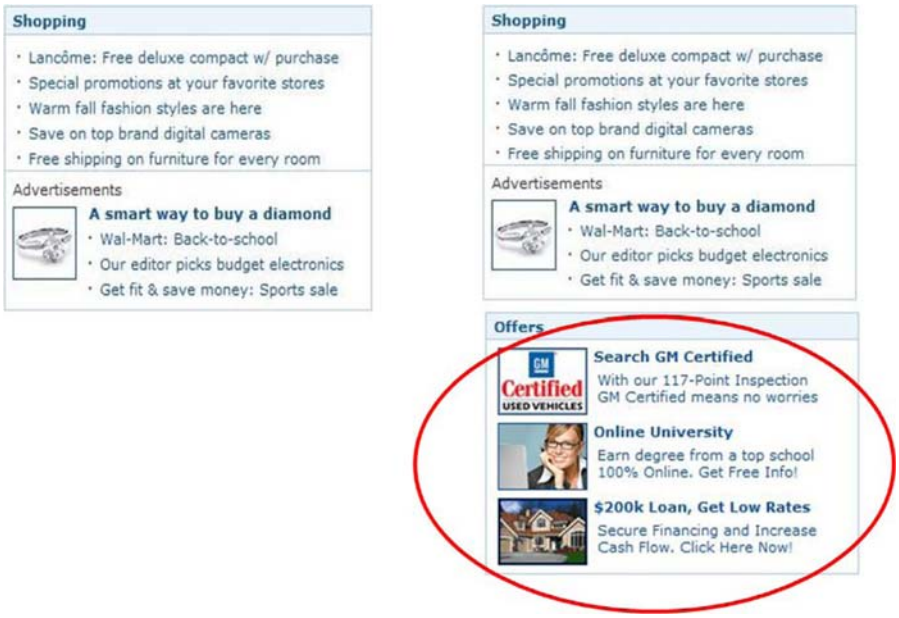


Fig. 5 MSN home page proposal. Left: Control, Right: proposed treatment

monetary value to each. (No statistically significant change was seen in visit frequency for this experiment.) Page views of the MSN home page have an assigned value based on ads; clicks to destinations from the MSN home page were estimated in two ways:

1. Monetary value that the destination property assigned to a click from the MSN home page. These destination properties are other sites in the MSN network. Such a click generates a visit to an MSN property (e.g., MSN Autos or MSN Money), which results in multiple page views.
2. The cost paid to search engines for a click that brings a user to an MSN property but not via the MSN home page (Search Engine Marketing). If the home page is driving less traffic to the properties, what is the cost of regenerating the "lost" traffic?

As expected, the number from #2 (SEM) was higher, as additional value beyond direct monetization is assigned to a click that may represent a new user, but the numbers were close enough to get agreement on the monetization value to use.

A controlled experiment was run on 5% of the MSN US home page users for 12 days. Clickthrough rate decreased by 0.38% (relative change), and the result was statistically significant (p-value $= 0.02$).

Translating the lost clicks to their monetary value, it was higher than the expected ad revenue, so the idea of adding more ads to the MSN home page was scrapped.

2.4 Behavior-Based Search at Amazon

The examples above changed User-Interface (UI) elements. This example deals with a backend algorithmic change, which is often overlooked as an area to apply controlled experiments.

Back in 2004, when several of the authors were in the Data Mining and Personalization department at Amazon, there already existed a good algorithm for making recommendations based on two sets. The signature feature for Amazon's recommendation is "People who bought item X bought item Y," but this was generalized to "People who ***viewed*** item X bought item Y" and "People who viewed item X ***viewed*** item Y." A proposal was made to use the same algorithm for "People who ***searched*** for X bought item Y." We called it Behavior-Based Search (BBS). In fact, the idea was to surface this in search results with no visible changes to the user interface. If a user searched for a string that was common, and there was a strong signal that people who searched for that string bought one of several items, these items would surface at the top of the search results. Note that this algorithm has no semantic understanding of the searched phrase, which was its strength and weakness.

Proponents of the algorithm gave examples of underspecified searches, such as "24," which most humans associated with the TV show starring Kiefer Sutherland. Amazon's search was returning poor results, shown in Fig. 6, such as CDs with 24 Italian Songs, clothing for 24-month old toddlers, a 24-inch towel bar, etc. (These results are still visible on Amazon today if you add an advanced search qualifier like "-foo" to the search phrase since this makes the search phrase unique and no mappings will exist from people who searched for it to products.) The BBS algorithm gave top-notch results with the DVDs of the show and with related books, i.e., things that people purchased after searching for "24" as shown in Fig. 6. The weakness of the algorithm was that some items surfaced that did not contain the words in the search phrase. For example, if one searches for "Sony HD DVD Player" (this example is recent as of January 2008), Toshiba HD DVDs will show up fairly high. The reason is that Sony makes Blu-Ray DVD players, not HD players, and that many users who search for Sony HD DVD players end up purchasing a Toshiba player. Given the pros and cons for the idea of Behavior-Based search, Amazon ran a controlled experiment.

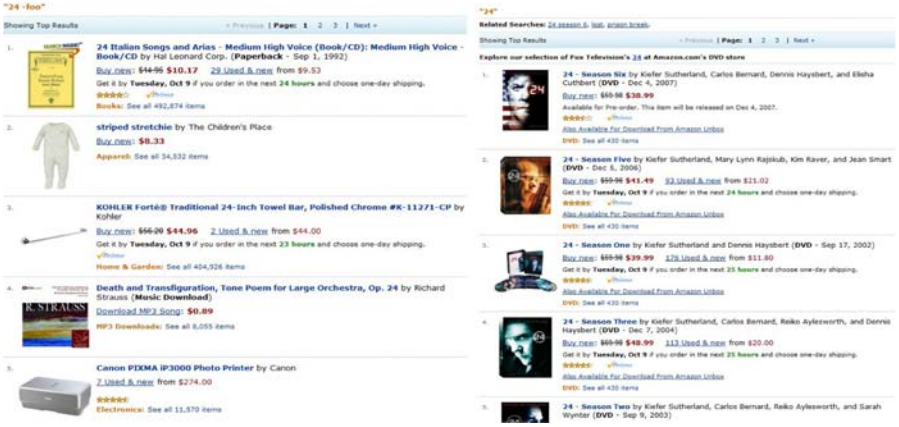


Fig. 6 Amazon search for "24" with and without BBS

In a UW iEdge Seminar talk by Amazon in April 2006, it was disclosed that the feature increased Amazon's revenue by 3%, which translates into several hundreds of millions of dollars.

2.5 Other examples

While these are extreme examples that are surprising in the magnitude of the difference, they show how hard it is to predict the success of new designs. Several more examples can be found in the emetrics talk on controlled experiments (Kohavi 2007).

Great examples of experiments are available at Marketing Experiments journal (McGlaughlin 2006), Design Choices Can Cripple a Website (Usborne 2005), Call to Action (Eisenberg and Eisenberg 2005), and Which Sells Best (Eisenberg and Garcia 2006). Forrester's Primer on A/B Testing (Chatham et al. 2004) mentions a few good examples of positive ROI:

- Marriott realized an additional $30 million in bookings with a new online reservations form.
- Luxury accessories retailer Coach improved the effectiveness of its site's search engine 200%—by having its vendor prove that a new search engine would produce more effective results with an A/B test.
- Disk-drive maker Iomega needed to know whether its prospects favored limited freeware or trial versions of full software products, and which email landing pages would produce the best conversion rate. Their solution? To structure an experimental design to test the various permutations—that ultimately drove up campaign yield by 50%.

Spool (2004) quantifies the cost of frustration at Amtrak.com's web site, by noting that it is very difficult to register and that only one out of every four attempts succeeds. Obviously making the site more usable will not increase registrations by a factor of three or four, but if registrations increased by 20%, he shows that revenues would increase by over $15M/year, enough to get someone's attention.

A/B test at InterContinental Hotels led the firm to add the range of available rates to search results, which added $45M–$60M of increased bookings (Manning et al. 2006).

In shop.com's The State of Retailing (Forrester Research 2005), the authors wrote that in their survey of 137 US retailers "100% of the retailers that employed usability testing and A/B testing of offers and promotions rank these tactics as effective or very effective."

Forrester's Web Analytics Spending Trends 2007 (Burns 2006) wrote that A/B testing will see the highest percentage of large increases [of web analytics categories]. A/B testing was one of only two categories [the other is SEO/SEM] in their survey that saw an increase in the percentage of respondents planning major budget growth.

3 Controlled experiments

Enlightened trial and error outperforms the planning of flawless execution
– David Kelly, founder of Ideo

To have a great idea, have a lot of them
– Thomas A. Edison

In the simplest controlled experiment, often referred to as an A/B test, users are randomly exposed to one of two variants: Control (A), or Treatment (B) as shown in Fig. 7 (Mason et al. 1989; Box et al. 2005; Keppel et al. 1992).

The key here is "random." Users cannot be distributed "any old which way" (Weiss 1997); no factor can influence the decision. Based on observations collected, an Overall Evaluation Criterion (OEC) is derived for each variant (Roy 2001).

For example, in Checkout Example (Sect. 2.1), the OEC can be the conversion rate, units purchased, revenue, profit, expected lifetime value, or some weighted combination of these. Analysis is then done to determine if the difference in the OEC for the variants is statistically significant.

If the experiment was designed and executed properly, the only thing consistently different between the two variants is the change between the Control and Treatment, so any differences in the OEC are inevitably the result of this assignment, establishing causality (Weiss 1997, p. 215).

There are several primers on running controlled experiments on the web (Peterson 2004, pp. 76–78; Eisenberg and Eisenberg 2005, pp. 283–286; Chatham et al. 2004; Eisenberg 2005, 2004; Quarto-vonTivadar 2006; Miller 2007, 2006; Kaushik 2006; Peterson 2005, pp. 248–253; Tyler and Ledford 2006, pp. 213–219; Sterne 2002, pp. 116–119).

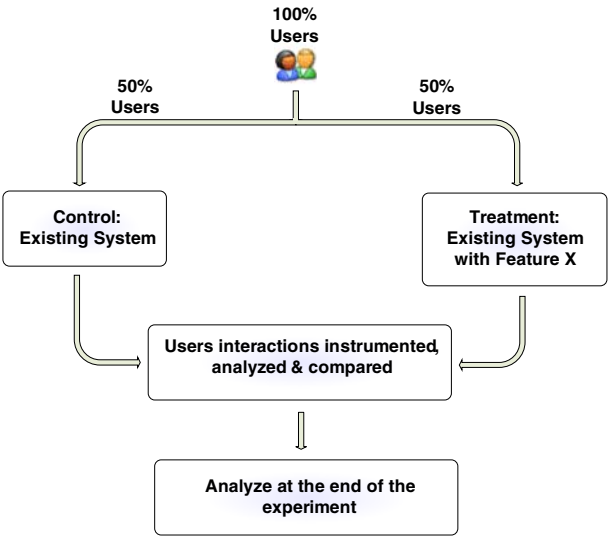


Fig. 7 High-level flow for an A/B test

While the concept is easy to understand and basic ideas echo through many references, there are important lessons that we share here that are rarely discussed. These will help experimenters understand the applicability, limitations, and how to avoid mistakes that invalidate the results.

3.1 Terminology

The terminology for controlled experiments varies widely in the literature. Below we define key terms used in this paper and note alternative terms that are commonly used.

Overall Evaluation Criterion (OEC) (Roy 2001). A quantitative measure of the experiment's objective. In statistics this is often called the *Response* or *Dependent Variable* (Mason et al. 1989; Box et al. 2005); other synonyms include *Outcome*, *Evaluation metric*, *Performance metric*, or *Fitness Function* (Quarto-vonTivadar 2006). Experiments may have multiple objectives and a scorecard approach might be taken (Kaplan and Norton 1996), although selecting a single metric, possibly as a weighted combination of such objectives is highly desired and recommended (Roy 2001, p. 50). A single metric forces tradeoffs to be made once for multiple experiments and aligns the organization behind a clear objective. A good OEC should not be short-term focused (e.g., clicks); to the contrary, it should include factors that predict long-term goals, such as predicted lifetime value and repeat visits. Ulwick describes some ways to measure what customers want (although not specifically for the web) (Ulwick 2005).

Factor. A controllable experimental variable that is thought to influence the OEC. Factors are assigned *Values*, sometimes called *Levels* or *Versions*. Factors are sometimes called *Variables*. In simple A/B tests, there is a single factor with two values: A and B.

Variant. A user experience being tested by assigning levels to the factors; it is either the Control or one of the Treatments. Sometimes referred to as *Treatment*, although we prefer to specifically differentiate between the Control, which is a special variant that designates the existing version being compared against and the new Treatments being tried. In case of a bug, for example, the experiment is aborted and all users should see the Control variant.

Experimental unit. The entity over which metrics are calculated before averaging over the entire experiment for each variant. Sometimes called an *item*. The units are assumed to be independent. On the web, the user is a common experimental unit, although some metrics may have user-day, user-session or page views as the experimental units. For any of these randomization by user is preferred. It is important that the user receive a consistent experience throughout the experiment, and this is commonly achieved through randomization based on user IDs stored in cookies. We will assume that randomization is by user with some suggestions when randomization by user is not appropriate in Appendix.

Null hypothesis. The hypothesis, often referred to as H_0, that the OECs for the variants are not different and that any observed differences during the experiment are due to random fluctuations.

Confidence level. The probability of failing to reject (i.e., retaining) the null hypothesis when it is true.

Power. The probability of correctly rejecting the null hypothesis, H_0, when it is false. Power measures our ability to detect a difference when it indeed exists.

A/A test. Sometimes called a Null Test (Peterson 2004). Instead of an A/B test, you exercise the experimentation system, assigning users to one of two groups, but expose them to exactly the same experience. An A/A test can be used to (i) collect data and assess its variability for power calculations, and (ii) test the experimentation system (the Null hypothesis should be rejected about 5% of the time when a 95% confidence level is used).

Standard deviation (Std-Dev). A measure of variability, typically denoted by σ.

Standard error (Std-Err). For a statistic, it is the standard deviation of the sampling distribution of the sample statistic (Mason et al. 1989). For a mean of n independent observations, it is $\hat{\sigma}/\sqrt{n}$ where $\hat{\sigma}$ is the estimated standard deviation.

3.2 Hypothesis testing and sample size

To evaluate whether one of the treatments is different than the Control, a statistical test can be done. We accept a Treatment as being statistically significantly different if the test rejects the null hypothesis, which is that the OECs are not different.

We will not review the details of the statistical tests, as they are described very well in many statistical books (Mason et al. 1989; Box et al. 2005; Keppel et al. 1992).

What is important is to review the factors that impact the test:

1. *Confidence level.* Commonly set to 95%, this level implies that 5% of the time we will incorrectly conclude that there is a difference when there is none (Type I error). All else being equal, increasing this level reduces our power (below).
2. *Power.* Commonly desired to be around 80–95%, although not directly controlled. If the Null Hypothesis is false, i.e., there is a difference in the OECs, the power is the probability of determining that the difference is statistically significant. (A Type II error is one where we retain the Null Hypothesis when it is false.)
3. *Standard error.* The smaller the Std-Err, the more powerful the test. There are three useful ways to reduce the Std-Err:
    a. The estimated OEC is typically a mean of large samples. As shown in Sect. 3.1, the Std-Err of a mean is inversely proportional to the square root of the sample size, so increasing the sample size, which usually implies running the experiment longer, reduces the Std-Err and hence increases the power for most metrics. See the example in 3.2.1.
    b. Use OEC components that have inherently lower variability, i.e., the Std-Dev, σ, is smaller. For example, conversion probability (0–100%) typically has lower Std-Dev than number of purchase units (typically small integers), which in turn has a lower Std-Dev than revenue (real-valued). See the example in 3.2.1.
    c. Lower the variability of the OEC by filtering out users who were not exposed to the variants, yet were still included in the OEC. For example, if you make a change to the checkout page, analyze only users who got to the page, as everyone else adds noise, increasing the variability. See the example in 3.2.3.

4. *Effect*. The difference in OECs for the variants, i.e. the mean of the Treatment minus the mean of the Control. Larger differences are easier to detect, so great ideas will unlikely be missed. Conversely, Type II errors are more likely when the effects are small.

Two formulas are useful to share in this context. The first is the *t*-test, used in A/B tests (single factor hypothesis tests):

$$t = \frac{\overline{O_B} - \overline{O_A}}{\widehat{\sigma_d}} \tag{1}$$

where $\overline{O_A}$ and $\overline{O_B}$ are the estimated OEC values (e.g., averages), $\widehat{\sigma_d}$ is the estimated standard deviation of the difference between the two OECs, and t is the test result. Based on the confidence level, a threshold t is established (e.g., 1.96 for large samples and 95% confidence) and if the absolute value of t is larger than the threshold, then we reject the Null Hypothesis, claiming the Treatment's OEC is therefore statistically significantly different than the Control's OEC. We assume throughout that the sample sizes are large enough that it is safe to assume the means have a Normal distribution by the Central Limit Theorem (Box et al. 2005, p. 29; Boos and Hughes-Oliver 2000) even though the population distributions may be quite skewed.

A second formula is a calculation for the minimum sample size, assuming the desired confidence level is 95% and the desired power is 80% (van Belle 2002, p. 31)

$$n = \frac{16\sigma^2}{\Delta^2} \tag{2}$$

where n is the number of users in each variant and the variants are assumed to be of equal size, σ^2 is the variance of the OEC, and Δ is the sensitivity, or the amount of change you want to detect. (It is well known that one could improve the power of comparisons of the treatments to the control by making the sample size of the control larger than for the treatments when there is more than one treatment and you are only interested in the comparison of each treatment to the control. If, however, a primary objective is to compare the treatments to each other then all groups should be of the same size as given by Formula 2.) The coefficient of 16 in the formula provides 80% power, i.e., it has an 80% probability of rejecting the null hypothesis that there is no difference between the Treatment and Control if the true mean is different than the true Control by Δ. Even a rough estimate of standard deviation in Formula 2 can be helpful in planning an experiment. Replace the 16 by 21 in the formula above to increase the power to 90%.

A more conservative formula for sample size (for 90% power) has been suggested (Wheeler 1974):

$$n = (4r\sigma/\Delta)^2 \tag{3}$$

where r is the number of variants (assumed to be approximately equal in size). The formula is an approximation and intentionally conservative to account for multiple

comparison issues when conducting an analysis of variance with multiple variants per factor (Wheeler 1975; van Belle 2002). The examples below use the first formula.

3.2.1 Example: impact of lower-variability OEC on the sample size

Suppose you have an e-commerce site and 5% of users who visit during the experiment period end up purchasing. Those purchasing spend about $75. The average user therefore spends $3.75 (95% spend $0). Assume the standard deviation is $30. If you are running an A/B test and want to detect a 5% change to revenue, you will need over 409,000 users to achieve the desired 80% power, based on the above formula: $16 * 30^2/(3.75 * 0.05)^2$.

If, however, you were only looking for a 5% change in conversion rate (not revenue), a lower variability OEC based on point 3.b can be used. Purchase, a conversion event, is modeled as a Bernoulli trial with $p = 0.05$ being the probability of a purchase. The standard deviation of a Bernoulli is $\sqrt{p(1-p)}$ and thus you will need less than 122,000 users to achieve the desired power based on $16 * (0.05 \cdot (1 - 0.05))/(0.05 \cdot 0.05)^2$.

Using conversion as the OEC instead of purchasing spend can thus reduce the sample size required for the experiment by a factor of 3.3. Because the number of site visitors is approximately linear in the running time for the experiment (the number of distinct users is sublinear due to repeat visitors, but a linear approximation is reasonable for most sites), this can reduce the running time of the experiment from 6 weeks to 2 weeks, and thus is worth considering.

3.2.2 Example: impact of reduced sensitivity on the sample size

Because the sensitivity, Δ, is squared in the formula for sample size, if the desired sensitivity is reduced to support detecting a 20% change in conversion instead of 5% (a factor of 4), the number of users needed drops by a factor of 16 to 7,600.

As will be discussed later on, this is the reason that detecting a bug in the implementation can be done quickly. Suppose you plan an experiment that will allow detecting a 1% change in the OEC, but a bug in the implementation exposes users to a bad experience and causes the OEC to drop by 20%. Such a bug can be detected not in 1/20th of the planned running time, but in 1/400th of the running time. If the experiment was planned to run for two weeks, you can detect an egregious problem in the first hour!

3.2.3 Example: filtering users not impacted by the change

If you made a change to the checkout process, you should only analyze users who started the checkout process (point 3.c), as others could not see any difference and therefore just add noise. Assume that 10% of users initiate checkout and that 50% of those users complete it. This user segment is more homogenous and hence the OEC has lower variability. Using the same numbers as before, the average conversion rate is 0.5, the std-dev is 0.5, and thus you will need only 6,400 users going through checkout to detect a 5% change based on $16 * (0.5(1 - 0.5))/(0.5 \cdot 0.05)^2$. Since we excluded the 90% who do not initiate, the total number of users to the website should be 64,000,

which is almost half the previous result of 122,000, thus the experiment could run for half the time and yield the same power.

3.2.4 The choice of OEC must be made in advance

When running experiments, it is important to decide in advance on the OEC (a planned comparison); otherwise, there is an increased risk of finding what appear to be significant results by chance (familywise type I error) (Keppel et al. 1992). Several adjustments have been proposed in the literature (e.g., Fisher's least-significant-difference, Bonferroni adjustment, Duncan's test, Scheffé's test, Tukey's test, and Dunnett's test), but they basically equate to increasing the 95% confidence level and thus reducing the statistical power (Mason et al. 1989; Box et al. 2005; Keppel et al. 1992).

3.3 Confidence intervals for absolute and percent effect

It is useful to give a confidence interval for the difference in the means of the Treatment and Control in addition to the results of the hypothesis test. The confidence interval gives a range of plausible values for the size of the effect of the Treatment whereas the hypothesis test only determines if there is a statistically significant difference in the mean.

3.3.1 Confidence intervals for absolute effect

The formula for the confidence interval for the difference in two means is fairly straightforward. Using the notation developed previously, the upper and lower bounds for a 95% confidence interval are

$$\text{CI Limits} = \overline{O_B} - \overline{O_A} \pm 1.96 * \widehat{\sigma_d} \tag{4}$$

One could use the confidence interval for the absolute affect to conduct a hypothesis test—if zero is in the interval you would not reject H_0, otherwise reject H_0 and conclude the Treatment has an effect.

3.3.2 Confidence intervals for percent effect

For many online metrics, the difference in the means is so small that *percent* change has much more intuitive meaning than the absolute difference. For example, for a recent experiment, the treatment effect for specific clickthrough rate was 0.00014. This translated to a 12.85% change due to the Treatment. The latter number was much more meaningful to decision makers. The percent difference is calculated by

$$\text{Pct Diff} = \frac{\overline{O_B} - \overline{O_A}}{\overline{O_A}} * 100\% \tag{5}$$

However, forming a confidence interval around the percent change is not a straightforward extension of the confidence interval for the absolute effect. This is because we are now dividing by a random variable. The initial derivation of this interval is due to Fieller (Willan and Briggs 2006). Note that if the denominator is stochastically close to zero one or both endpoints will not exist. In practice, you shouldn't calculate this interval if the confidence interval for the denominator contains zero.

Define the coefficient of variation of the two groups to be

$$CV_B = \frac{\widehat{\sigma_B}}{\overline{O_B}}$$
$$CV_B = \frac{\widehat{\sigma_A}}{\overline{O_A}}$$

The lower and upper bounds for a 95% confidence interval for the percent difference are

$$\text{CI for Percent Effect} = (\text{PctDiff} + 1) * \frac{1 \pm 1.96 * \sqrt{CV_A^2 + CV_B^2 - (1.96^2) * CV_A^2 * CV_B^2}}{1 - (1.96) * CV_A^2} - 1 \quad (6)$$

These formulas assume the covariance between the Treatment and Control mean is zero which will be true in a controlled experiment when the randomization is carried out properly.

3.4 Effect of robots on experimental results

Robots can introduce significant skew into estimates, enough to render assumptions invalid. We have seen cases where robots caused many metrics to be significant when they should not have been (e.g., much more than 5% false positives for an A/A test). For the purpose of experimentation, it is especially important to remove some types of robots, those that interact with the user-id. For some websites robots are thought to provide up to half the pageviews on the site(Kohavi et al. 2004). Since many robots have the same characteristics as human users it is difficult to clearly delineate between the two. Benign or simple robots can often be filtered by basic characteristics (e.g. user agent, IP address) but many modern robots use sophisticated techniques to escape detections and filtering (Tan and Kumar 2002).

3.4.1 JavaScript versus server-side call

It is generally thought that very few robots will be included in the experiment if the treatment assignment is called by JavaScript so those experimental setups shouldn't be affected as much by robots. This should be validated by the experimenter.

3.4.2 Robots that reject cookies

We recommend excluding unidentified requests from the analysis, so that robots that reject cookies will not be part of the experimental results. If the treatment assignment and data collection is based only on users with a user ID stored in the user's cookie, these robots will not be counted in the number of users or in the data that is collected on user behavior.

3.4.3 Robots that accept cookies

If a robot accepts cookies and does not delete them, the effect can be profound, especially if the robot has a large number of actions on the site. We have found that there are usually a relatively small number of these robots but their presence in the Treatment or Control can seriously bias the comparison. For example, we have found some robots that have up to 7,000 clicks on a page in an hour or more than 3,000 page views in a day. Any hypothesis test comparing Treatment and Control when these robots are present can be very misleading. These robots will not only bias the estimate of the effect, they also increase the standard deviation of many metrics, thus reducing the power.

Therefore, we need to aggressively filter out robots that do not delete cookies and have a large number of actions (e.g. pageviews or clickthroughs (triggered by onclick JavaScript handlers)) for a single user-id. Robots that either do not accept cookies or clear cookies after one or only a few actions will not have much of an effect on the comparison of Treatment to Control. Robot filtering can be accomplished through a combination of omitting users whose user agent is on a list of known robots and through the use of heuristics (Kohavi 2003). The heuristics may vary depending on the website.

3.5 Extensions for online settings

Several extensions to basic controlled experiments are possible in an online setting (e.g., on the web).

3.5.1 Treatment ramp-up

An experiment can be initiated with a small percentage of users assigned to the Treatment(s), and then that percentage can be gradually increased. For example, if you plan to run an A/B test at 50%/50%, you might start with a 99.9%/0.1% split, then rampup the Treatment from 0.1% to 0.5% to 2.5% to 10% to 50%. At each step, which could run for, say, a couple of hours, you can analyze the data to make sure there are no egregious problems with the Treatment before exposing it to more users. The square factor in the power formula implies that such errors could be caught quickly on small populations and the experiment can be aborted before many users are exposed to the bad Treatment.

3.5.2 Automation

Once an organization has a clear OEC, it can run experiments to optimize certain areas amenable to automated search. For example, the slots on the home page at Amazon are automatically optimized (Kohavi et al. 2004). If decisions have to be made quickly (e.g., headline optimizations for portal sites), these could be made with lower confidence levels because the cost of mistakes is lower. Multi-armed bandit algorithms (Wikepedia 2008) and Hoeffding Races (Maron and Moore 1994) can be used for such optimizations.

3.5.3 Software migrations

Experiments can be used to help with software migration. If a feature or a system is being migrated to a new backend, new database, or a new language, but is not expected to change user-visible features, an A/B test can be executed with the goal of retaining the Null Hypothesis, which is that the variants are not different. We have seen several such migrations, where the migration was declared complete, but an A/B test showed significant differences in key metrics, helping identify bugs in the port. Because the goal here is to retain the Null Hypothesis, it is crucial to make sure the experiment has enough statistical power to actually reject the Null Hypothesis if it false.

3.6 Limitations

Despite significant advantages that controlled experiments provide in terms of causality, they do have limitations that need to be understood. Some, which are noted in the Psychology literature are not relevant to the web (Rossi et al. 2003, pp. 252–262; Weiss 1997), but some limitations we encountered are certainly worth noting.

1. *Quantitative metrics, but no explanations*. It is possible to know which variant is better, and by how much, but not "why." In user studies, for example, behavior is often augmented with users' comments, and hence usability labs can be used to augment and complement controlled experiments (Nielsen 2005).
2. *Short term versus long term effects*. Controlled experiments measure the effect on the OEC during the experimentation period, typically a few weeks. While some authors have criticized that focusing on a metric implies short-term focus (Quarto-vonTivadar 2006; Nielsen 2005), we disagree. Long-term goals *should* be part of the OEC. Let us take search ads as an example. If your OEC is revenue, you might plaster ads over a page, but we know that many ads hurt the user experience, so a good OEC should include a penalty term of usage of real-estate for ads that are not clicked, and/or should directly measure repeat visits and abandonment. Likewise, it is wise to look at delayed conversion metrics, where there is a lag from the time a user is exposed to something and take action. These are sometimes called latent conversions (Miller 2006; Quarto-vonTivadar 2006). Coming up with good OECs is hard, but what is the alternative? The key point here is to recognize this limitation, but avoid throwing the baby out with the bathwater.

3. *Primacy and newness effects*. These are opposite effects that need to be recognized. If you change the navigation on a web site, experienced users may be less efficient until they get used to the new navigation, thus giving an inherent advantage to the Control. Conversely, when a new design or feature is introduced, some users will investigate it, click everywhere, and thus introduce a "newness" bias. This bias is sometimes associated with the Hawthorne effect (2007). Both primacy and newness concerns imply that some experiments need to be run for multiple weeks. One analysis that can be done is to compute the OEC only for new users on the different variants, since they are not affected by either factor.
4. *Features must be implemented*. A live controlled experiment needs to expose some users to a Treatment different than the current site (Control). The feature may be a prototype that is being tested against a small portion, or may not cover all edge cases (e.g., the experiment may intentionally exclude 20% of browser types that would require significant testing). Nonetheless, the feature must be implemented and be of sufficient quality to expose users to it. Nielsen (2005) correctly points out that paper prototyping can be used for qualitative feedback and quick refinements of designs in early stages. We agree and recommend that such techniques complement controlled experiments.
5. *Consistency*. Users may notice they are getting a different variant than their friends and family. It is also possible that the same user will see multiple variants when using different computers (with different cookies). It is relatively rare that users will notice the difference.
6. *Parallel experiments*. Our experience is that strong interactions are rare in practice (van Belle 2002), and we believe this concern is overrated. Raising awareness of this concern is enough for experimenters to avoid tests that can interact. Pairwise statistical tests can also be done to flag such interactions automatically.
7. *Launch Events and Media Announcements*. If there is a big announcement made about a new feature, such that the feature is announced to the media, all users need to see it.

4 MultiVariable Testing[1]

An experiment that includes more than one factor is often called a MultiVariable test (MVT) (Alt and Usborne 2005). For example, consider testing five factors on the MSN homepage in a single experiment. A screenshot of the MSN homepage showing the control for each of these factors is given in Fig. 8.

[1] This is also known as Multivariate testing. We use the term MultiVariable Testing for two reasons. These tests were first called MultiVariable Tests in 1996 in an article in Forbes (Koselka 1996) referring to designed experiments in areas including sales and marketing. In addition, these tests are part of the statistical literature in the Design of Experiments field. There is a separate field of statistics known as multivariate statistics that does not deal with this topic so using the term multivariate could be a source of confusion.

Factor	Control	Treatment
F1	Shopping module as above	Add Offers module below
F2	Shopping module as above	Red border and heading
F3	Money and Quotes as above	Merge into one Money module
F4	Static ad shown to everyone	Ad shown depends on recent user behavior
F5	Video headlines chosen by editors	Order of headlines chosen by popularity/competition

In a single test we can estimate the (main) effects of each factor as well as the interactive effects between factors. First, we will consider the benefits and limitations of MVT versus one-factor-at-a-time, or A/B tests. Then we will discuss three approaches to online MVTs and how each approach takes advantage of the potential benefits and mitigates the limitations.

There are two primary benefits of a single MVT versus multiple sequential A/B tests to test the same factors:

1. You can test many factors in a short period of time, accelerating improvement. For example, if you wanted to test five changes to the website and you need to run each A/B test four weeks to achieve the power you need, it will take at least five

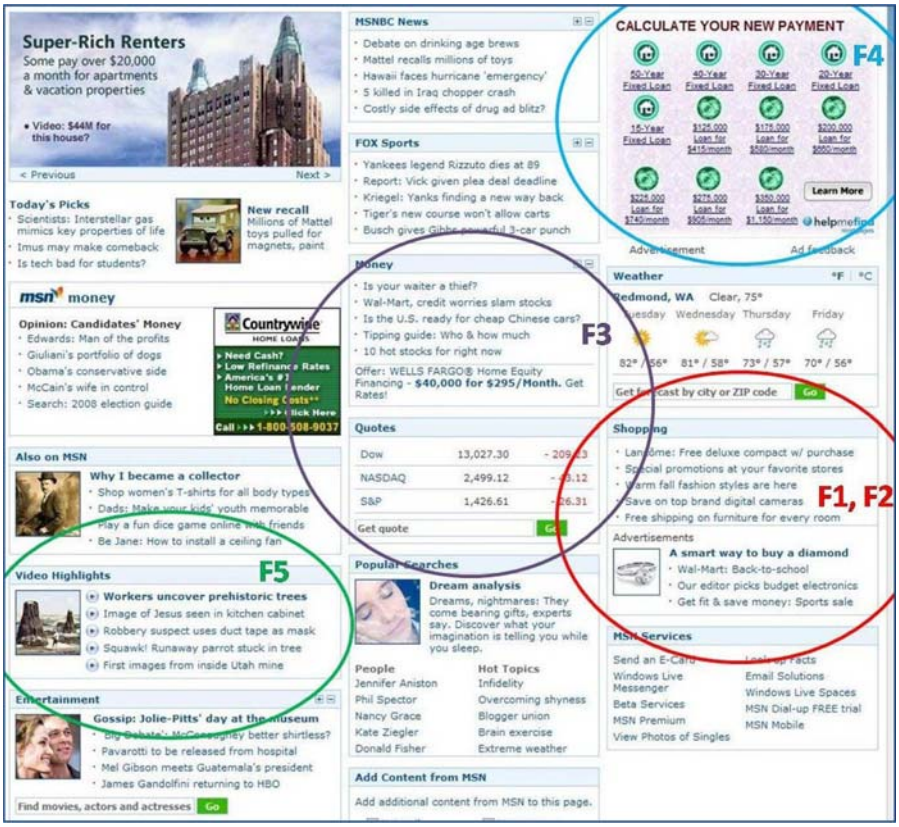


Fig. 8 Screenshot of MSN Homepage with controls for the five factors

months to complete the A/B tests. However, you could run a single MVT with all five factors in one month with the same power as with the five A/B tests.
2. You can estimate interactions between factors. Two factors interact if their combined effect is different from the sum of the two individual effects. If the two factors work together to enhance the outcome the interaction is synergistic. If instead they work against each other to dampen the effect, the interaction is antagonistic.

Three common limitations are:

1. Some combinations of factors may give a poor user experience. For example, two factors being tested for an online retailer may be enlarging a product image or providing additional product detail. Both may improve sales when tested individually, but when both are done at the same time the "buy box" is pushed below the fold and sales decrease. This would be a large antagonistic interaction. This interaction should be caught in the planning phase so that these two factors would not be tested at the same time.
2. Analysis and interpretation are more difficult. For a single factor test you typically have many metrics for the Treatment-Control comparison. For an MVT you have the same metrics for many Treatment-Control comparisons (at least one for each factor being tested) plus the analysis and interpretation of the interactions between the factors. Certainly, the information set is much richer but it can make the task of assessing which treatments to roll out more complex.
3. It can take longer to begin the test. If you have five factors you want to test and plan to test them one at a time you can start with any of those that are ready to be tested and test the others later. With an MVT you must have all five ready for testing at the beginning of the test. If any one is delayed, this would delay the start of the test.

We don't believe any of the limitations are serious ones in most cases, but they should be recognized before conducting an MVT. Generally, we believe the first experiment one does should be an A/B test mainly due to the complexity of testing more than one factor in the same test.

There are three overarching philosophies to conducting MVTs with online properties.

4.1 Traditional MVT

This approach uses designs that are used in manufacturing and other offline applications. These designs are most often fractional factorial (Davies and Hay 1950) and Plackett and Burman (1946) designs that are specific subsets of full factorial designs (all combinations of factor levels). These designs were popularized by Genichi Taguchi and are sometimes known as Taguchi designs. The user must be careful to choose a design that will have sufficient resolution to estimate the main effects and interactions that are of interest.

For our MSN example we show designs for a test of these five factors with a full factorial, a fractional factorial or a Plackett-Burman design.

Table 1 Fractional factorial design to test five factors with eight user groups

User groups	Factor levels assigned to each group				
	F1	F2	F3	F4	F5
1	−1	−1	−1	1	1
2	−1	−1	1	1	−1
3	−1	1	−1	−1	1
4	−1	1	1	−1	−1
5	1	−1	−1	−1	−1
6	1	−1	1	−1	1
7	1	1	−1	1	−1
8	1	1	1	1	1

Full factorial has all combinations of the factors which would be $2^5 = 32$ user groups. A fractional factorial is a fraction of the full factorial that has 2^K user groups and each column is orthogonal to the other four columns. There are obviously many such fractions with 8 and 16 user groups. One fractional factorial for $K = 3$ is given in Table 1 where −1 denotes the control and 1 denotes the treatment.

Plackett–Burman designs can be constructed where the factors are all at two levels with the number of user groups being a multiple of 4, so 4, 8, 12, 16, 20, etc. The number of factors that can be tested for any of these designs is the number of user groups minus one. If the number of user groups is a power of two the Plackett–Burman design is also a fractional factorial.

As with the fractional factorials, there are usually many Plackett–Burman designs that could be used for a given number of user groups.

In the statistical field of Design of Experiments, a major research area is to find designs that minimize the number of user groups needed for the test while allowing you to estimate the main effects and interactions with little or no confounding. The fractional factorial in Table 1 can estimate all five main effects but cannot estimate interactions well (Box et al. 2005, pp. 235–305). For many experimenters one of the primary reasons for running an MVT is to estimate the interactions among the factors being tested. You cannot estimate any interactions well with this design since all interactions are totally confounded with main effects or other two-factor interactions. No amount of effort at analysis or data mining will allow you to estimate these interactions individually. If you want to estimate all two factor interactions with five factors you will need a fractional factorial design with 16 treatment combinations. The Placket–Burman design in Table 2 has all two factor interactions partially confounded with main effects and other two factor interactions. This makes the estimation of these two factor interactions challenging.

We recommend two alternatives that we believe are better than the traditional MVT approach for online tests. The one you prefer will depend on how highly you value estimating interactions.

4.2 MVT by running concurrent tests

Fractions of the full factorial are used in offline testing because there is usually a cost to using more treatment combinations even when the number of experimental units

Table 2 Plackett–Burman design to test five factors with 12 user groups

User groups	Factor levels assigned to each group				
	F1	F2	F3	F4	F5
1	1	1	1	1	1
2	−1	1	−1	1	1
3	−1	−1	1	−1	1
4	1	−1	−1	1	−1
5	−1	1	−1	−1	1
6	−1	−1	1	−1	−1
7	−1	−1	−1	1	−1
8	1	−1	−1	−1	1
9	1	1	−1	−1	−1
10	1	1	1	−1	−1
11	−1	1	1	1	−1
12	1	−1	1	1	1

does not increase. This does not have to be the case with tests conducted with web sites. If we set up each factor to run as a one-factor experiment and run all these tests concurrently we can simplify our effort and get a full factorial in the end. In this mode we start and stop all these one-factor tests at the same time on the same set of users with users being independently randomly assigned to each experiment. The end result is you will have a full factorial experiment in all the factors you are testing. Of course, with a full factorial you will be able to estimate any interaction you want. A side benefit of this approach is that you can turn off any factor at any time (for example if a treatment for a factor is disastrous) without affecting the other factors. The experiment that includes the remaining factors is not affected.

It is commonly thought that the power of the experiment decreases with the number of treatment combinations (cells). This may be true if the analysis is conducted by comparing each individual cell to the Control cell. However, if the analysis is the more traditional one of calculating main effects and interactions using all the data for each effect, little or no power is lost. (A very slight loss of power could occur if one of the factors of combination of factors increases the variation of the response. Using a pooled estimate for experimental error will minimize any loss of power.) If your sample size (e.g. number of users) is fixed, it doesn't matter if you are testing a single factor or many or whether you are conducting an eight run MVT or a full factorial the power to detect a difference for any main effect is the same (Box et al. 2005). There **are** two things that will decrease your power, though. One is increasing the number of levels (variants) for a factor. This will effectively decrease the sample size for any comparison you want to make, whether the test is an MVT or an A/B test. The other is to assign less than 50% of the test population to the treatment (if there are two levels). It is especially important for treatments in an MVT to have the same percentage of the population as the Control.

4.3 Overlapping experiments

This approach is to simply test a factor as a one-factor experiment when the factor is ready to be tested with each test being independently randomized. It is distinct from the

previous Sect. (4.2) in that here each experiment turns on when the treatment is ready to go rather than launching all factors of a factorial design at once. In an agile software development world, there are significant benefits to running more tests frequently, as they are ready to be deployed. These tests can be going on simultaneously if there is no obvious user experience issue with the combinations that could be shown to any visitor. This is the approach you should take if you want to maximize the speed with which ideas are tested and you are not interested in or concerned with interactions. Large interactions between factors are actually rarer than most people believe, unless they are already known, such as with the buy box example. This is a much better alternative than the traditional approach mentioned first. With the traditional approach you have the limitation that you can't test until all the factors are ready to be tested. In addition, when you're done (with many test designs that are recommended) you won't be able to estimate interactions well if at all. With overlapping experiments you test the factors more quickly and, if there is sufficient overlap in any two factors, you can estimate the interaction between those factors. If you are especially interested in the interaction between two specific factors you can plan to test those factors at the same time.

We believe the two alternatives presented above are better than the traditional MVT approach for online experiments. The one you use would depend on your priorities. If you want to test ideas as quickly as possible and aren't concerned about interactions, use the overlapping experiments approach. If it is important to estimate interactions run the experiments concurrently with users being independently randomized into each test effectively giving you a full factorial experiment.

5 Implementation architecture

Implementing an experiment on a website involves three components. The first component is the *randomization algorithm*, which is a function that maps end users to variants. The second component is the *assignment method*, which uses the output of the randomization algorithm to determine the experience that each user will see on the website. The third component is the *data path*, which captures raw observation data as the users interact with the website, aggregates it, applies statistics, and prepares reports of the experiment's outcome.

5.1 Randomization algorithm

Finding a good *randomization algorithm* is critical because the statistics of controlled experiments assume that each variant of an experiment is assigned a random sample of end users. Randomization algorithms must have the following three properties to support statistically correct experiments (using the methodology presented above):

1. End users must be equally likely to see each variant of an experiment (assuming a 50–50 split). There should be no bias toward any particular variant.
2. Repeat assignments of a single end user must be *consistent*; the end user should be assigned to the same variant on each successive visit to the site.

3. When multiple experiments are run, there must be no correlation between experiments. An end user's assignment to a variant in one experiment must have no effect on the probability of being assigned to a variant in any other experiment.

Randomization algorithms may optionally support the following two desirable properties:

4. The algorithm may support *monotonic ramp-up*, meaning that the percentage of users who see a Treatment can be slowly increased without changing the assignments of users who were previously assigned to that Treatment. Supporting this property allows the Treatment percentage to be slowly increased without impairing the user experience or damaging the validity of the experiment.
5. The algorithm may support *external control*, meaning that users can be manually forced into and out of variants. This property makes it easier to test the experimental site.

The remainder of this section will only consider techniques that satisfy at least the first three properties.

5.1.1 Pseudorandom with caching

A standard pseudorandom number generator can be used as the randomization algorithm when coupled with a form of caching. A good pseudorandom number generator will, by itself, satisfy the first and third requirements of the randomization algorithm.

We tested several popular random number generators on their ability to satisfy the first and third requirements. We tested five simulated experiments against one million sequential user IDs, running chi-square tests to look for interactions. We found that the random number generators built into many popular languages (for example, C#) work well as long as the generator is seeded only once at server startup. Seeding the random number generator on each request may cause adjacent requests to use the same seed which may (as it did in our tests) introduce noticeable correlations between experiments. In particular, we found that the technique employed by Eric Peterson using Visual Basic (Peterson 2005) creates two-way interactions between experiments.

To satisfy the second requirement, the algorithm must introduce state: the assignments of end users must be cached once they visit the site. The caching can be accomplished either on the server side (by storing the assignments for all users in some form of database), or on the client side (by storing a user's assignment in a cookie).

The database approach is expensive (in hardware), and has space requirements that increase linearly with the number of experiments and the number of users. However, it easily satisfies the fifth property by allowing the user assignment database to be modified. The cookie approach is significantly cheaper, requiring no database and costing only a linear amount of space in the number of experiments (this time within the user's cookie). It will not work for users with cookies turned off.

Both forms of this approach are difficult to scale up to a large system with a large fleet of servers. The server making the random assignment must communicate its state to all the other servers (including those used for backend algorithms) in order

to keep assignments consistent. This sort of propagation is expensive and difficult to implement correctly.

The fourth requirement (monotonic ramp-up) is difficult to implement using this method, and so many systems ignore the requirement altogether. Regardless of which approach is used to maintain state, the system would need to carefully reassign Control users who visit the site after a ramp-up. The difficulty comes in determining the percentage Treatment to assign to these users so that the overall Treatment percentage reaches the desired value.

5.1.2 Hash and partition

This method eliminates the need for caching by replacing the random number generator with a *hash function*. Unlike a random number generator, which produces randomly distributed numbers independent of any input, a (good) hash function produces randomly distributed numbers as a function of a specific input. The method works as follows: each user is assigned a single unique identifier which is maintained either through a database or a cookie. Likewise, each experiment is assigned a unique identifier. A hash function is applied to the combination of the user identifier and the experiment identifier (e.g. by concatenating them together) to obtain an integer that is uniformly distributed on a range of values. The range is then partitioned, with each variant represented by a partition. The unique user identifier may be reused across any number of experiments.

This method is very sensitive to the choice of hash function. If the hash function has any *funnels* (instances where adjacent keys map to the same hash code) then the first property (uniform distribution) will be violated. And if the hash function has *characteristics* (instances where a perturbation of the key produces a predictable perturbation of the hash code), then correlations may occur between experiments. Few hash functions are sound enough to be used in this technique.

We tested this technique using several popular hash functions and a methodology similar to the one we used on the pseudorandom number generators. While any hash function will satisfy the second requirement (by definition), satisfying the first and third is more difficult. We tested five simulated experiments against one million sequential user IDs. We ran chi-square tests to look for violations of the first and third requirements of a randomization algorithm and found that only the cryptographic hash function MD5 generated no correlations between experiments. SHA256 (another cryptographic hash) came close, requiring a five-way interaction to produce a correlation. Other hash functions (including the string hashing algorithm built into .net) failed to pass even a two-way interaction test.

The running-time performance of the hash and partition approach is limited by the running-time performance of the hash function, which can be an issue because cryptographic hashes like MD5 are expensive and clever attempts to improve the performance through partial caching generally fail. For example, one system attempted to compute separate hashes of both the experiment name and the end user which would then be XOR'd together. The intent was to avoid the cost of MD5 by caching the hashed experiment name in the experiment, caching the hashed end user id in the user's cookie, and executing only the final XOR at assignment time. This technique produces severe

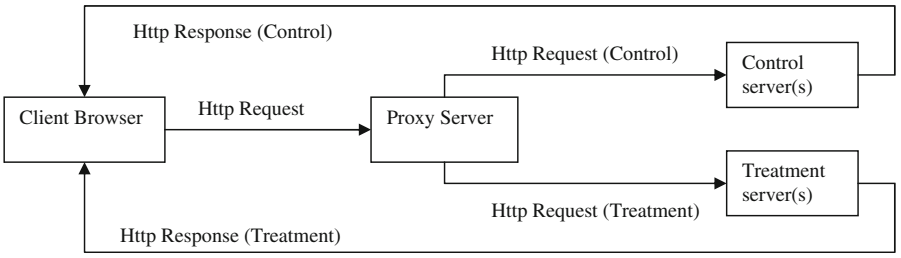
correlations between experiments: assuming two experiments with two variants each running at 50/50, if the most significant bit of the hashes of the experiment names for two experiments matched, users would always get the same assignment across both experiments. If they did not match, users would get exactly the opposite assignment between experiments. Either way, the third property is violated and results of both experiments are confounded.

Satisfying the fifth property (external control) is very difficult with a raw hash and partition approach. The simplest way to satisfy this property is to use a hybrid approach, combining the hash and partition method with either a small database or limited use of cookies. Because the set of users subject to external control is typically small (e.g. users designed by a test team), this hybrid approach should not encounter the full disadvantages of the pseudorandom with caching technique.

5.2 Assignment method

The assignment method is the piece of software that enables the experimenting website to execute a different code path for different end users. A good assignment method can manipulate anything from visible website content to backend algorithms. There are multiple ways to implement an assignment method. In the remainder of this section, we compare several common assignment methods and recommend best practices for their use.

5.2.1 Traffic splitting



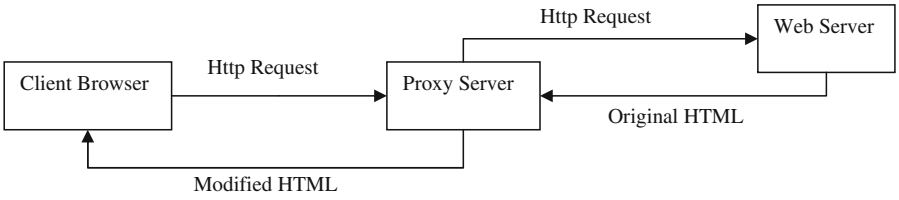
Traffic splitting refers to a family of assignment methods that involve implementing each variant of an experiment on a different logical fleet of servers. These can be different physical servers, different virtual servers, or event different ports on the same machine. The website uses either a load balancer or proxy server to split traffic between the variants and the randomization algorithm must be embedded at this level. Traffic splitting has the advantage of being *non-intrusive*; no changes to existing code are required to implement an experiment. However, the approach has significant disadvantages:

1. Running experiments on small features is disproportionately difficult because the entire application must be replicated regardless of the size of the change.
2. Setting up and configuring parallel fleets is typically expensive. The Control fleet must have sufficient capacity to take 100% of the traffic in the event that the

experiment needs to be shut down. The Treatment fleet(s) may be smaller, but their size will limit the maximum percentage that may be assigned to each Treatment.
3. Running multiple experiments requires the fleet to support one partition for each combination of variants across all experiments. This number increases as the number of tested combinations increases (potentially exponentially in the number of simultaneous experiments).
4. Any differences between the fleets used for each variant may confound the experimental results. Ideally, the hardware and network topology of each fleet will be identical and A/A tests will be run to confirm the absence of fleet-related effects.

The drawback of traffic splitting is that it is an expensive way to implement an experiment, even though the method appears cheap because it minimizes IT/developer involvement. We recommend this method for testing changes that introduce significantly different code, such as migration to a new website platform, the introduction of a new rendering engine, or a complete upgrade of a website.

5.2.2 Page rewriting



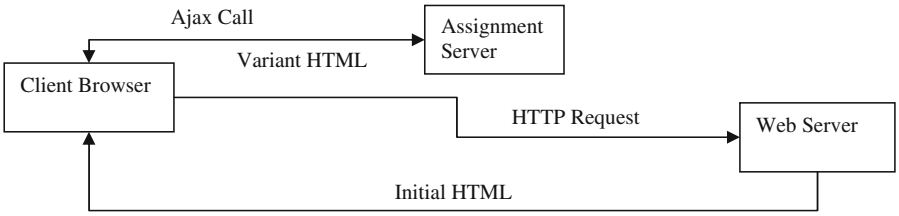
Page rewriting is an assignment method that incorporates a special type of proxy server that modifies HTML content before it is presented to the end user. Using this approach, the end-user's browser sends a request to the proxy server, which forwards it on to the experimenting website after recording some data. Then, the HTML response from the experimenting website passes back through the proxy server on its way to the end user's browser. The proxy server applies the randomization algorithm, selects variants for one or more experiments, and modifies the HTML according to the selected variants (e.g. by applying substitution rules expressed as regular expressions or XPath queries). The server then sends the modified HTML to the end user's browser. At least one commercial provider (SiteSpect 2008) offers a solution based on this method. Like traffic splitting, this method is *non-intrusive*. However, it still incurs some disadvantages:

1. Page render time is impacted by the action of the proxy server. Render time will be affected by both the time required to rewrite the HTML and the network latency between the proxy server and the web server.
2. Experimentation on large sites requires significant hardware. Because the proxy servers both need to handle all potential traffic to the site and may become a point of failure, a large number of servers may be required to ensure scalability and availability of the website.

3. Development and testing of variant content is more difficult and more error-prone than with other methods. Each variant must be expressed as a set of rules for modifying HTML code rather than as the HTML code itself.
4. Running experiments on backend algorithms is difficult because the assignment decision is made after the page is rendered by the website.
5. Running experiments on encrypted traffic (in particular, pages served via https) is resource-intensive because the proxy server must decrypt, modify, and re-encrypt the content. This represents a significant problem because the most interesting parts of a website (such as the checkout page) are commonly encrypted.

Page rewriting can be a cheap method for experimenting on front-end content because it minimizes IT/developer involvement. However, it is not appropriate for testing back-end changes or platform migrations.

5.2.3 Client-side assignment



Client-side page modification is the most popular assignment method found in third-party experimentation platforms. It is supported by numerous products including Google Website Optimizer (2008), Omniture's Offermatica (Omniture 2008), Interwoven's Optimost (2008), Widemile (2008), and Verster (2008).

All of these products can run an experiment without making any decisions on the server. A developer implements an experiment by inserting JavaScript code that instructs the end user's browser to invoke an assignment service at render time. The service call returns the appropriate variant for the end user, and triggers a JavaScript callback that instructs the browser to dynamically alter the page being presented to the user, typically by modifying the DOM. The modification must occur *before* any part of the page renders, so any latency in the service call will add to the overall page render time. The content for each variant can either be cleverly embedded into the page or can be served by the assignment service. This method, although intrusive, can be very easy to implement: all the developer needs to do is add a small snippet of JavaScript to a page. However, it has some key limitations:

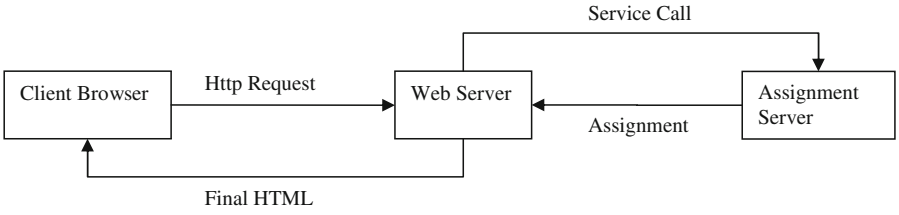
1. The client-side assignment logic executes after the initial page is served and therefore delays the end user experience, especially if the assignment service gets overloaded or if the end user is on a slow connection or is located far from the assignment server.
2. The method is difficult to employ on complex sites that rely on dynamic content because complex content can interact with the JavaScript code that modifies the page.

3. End users can determine (via the browser's View Source command) that a page is subject to experimentation, and may even (in some implementations) be able to extract the content of each variant.

Note: some implementations of this method attempt to optimize render time by avoid the service call if the end user is known (via cookie) to be in the Control. This optimization is incorrect (and should not be used) because it causes the render time delay to become correlated with variant assignment, thereby adding a confounding factor to the experiment.

This method is best for experiments on front-end content that is primarily static.

5.2.4 Server-side assignment



Server-side assignment refers to a family of methods that use code embedded into the website's servers to produce a different user experience for each variant. The code takes the form of an API call placed at the point where the website logic differs between variants. The API invokes the randomization algorithm and returns the identifier of the variant to be displayed for the current user. The calling code uses this information to branch to a different code path for each variant. The API call can be placed anywhere on the server side, in front-end rendering code, back-end algorithm code, or even in the site's content-management system. A complex experiment may make use of multiple API calls inserted into the code at different places. While the API can be implemented as a local function call, it typically uses an external service to ensure that the assignment logic stays consistent across a large server fleet. Server-side assignment is very *intrusive*; it requires deep changes to the experimenting application's code. Nonetheless, server-side assignment has three distinct advantages:

1. It is an extremely general method; it is possible to experiment on virtually any aspect of a page simply by modifying its code.
2. It places the experimentation code in the best logical place—right where decisions are made about a change. In particular, it is possible to experiment on backend features (for example, search and personalization algorithms) without touching the front end.
3. Experimentation is completely transparent to end users. End users experience only minimal delay and cannot discern that an experiment is running on the site.

Server-side assignment also has a number of disadvantages, all of which are stem from its intrusiveness:

1. Initial implementation is expensive. Depending on the complexity of the site, implementing the necessary server-side code changes can be difficult.
2. Because the method requires a developer to change code deep in the page logic for each experiment, implementing an experiment introduces risk. The risk is greatest on complex features whose code is spread across many pages and/or services.
3. Some variations of this method require code changes to be manually undone to complete an experiment. Specifically, a programmer must remove the code path that implements the losing treatment along with the conditional logic that reacts to the end user's treatment assignment. While this simply refers to code clean-up, leaving the losing treatment code in there can yield a very messy codebase, while removing it adds risk since production code will be modified. While this process is trivial for a simple one-page experiment, it can be a painful process if API calls are spread throughout the code, and all such code changes introduce additional risk.

Server-side assignment can be integrated into a content management system to greatly reduce the cost of running experiments using this method. When so integrated, experiments are configured by changing *metadata* instead of code. The metadata may be represented by anything from an editable configuration file to a relational database managed by a graphical user interface. The method is best illustrated with an example from a real system running at Amazon.com.

Amazon's home page is built on a content management system that assembles the page from individual units called *slots* (Kohavi et al. 2004). The system refers to page metadata at render time to determine how to assemble the page. Non-technical content editors schedule pieces of content in each slot through a graphical user interface that edits this page metadata. Content can include anything from an advertisement, to a product image, to a snippet of text filled with links, to a widget that displays dynamic content (such as personalized recommendations). A typical experiment would be to try various pieces of content in different locations. For example, do the recommendations receive higher clickthrough on the left or on the right? To enable this sort of experiment, the content management system is extended to allow pieces of content to be scheduled with respect to a specific experiment. As the page request comes in, the system executes the assignment logic for each scheduled experiment and saves the results to page context where the page assembly mechanism can react to it. The content management system only needs to be modified once; from then on, experiments can be designed, implemented, and removed by modifying the page metadata through the user interface.

5.2.5 Summary

The following table summarizes the relative advantages and disadvantages of all of the assignment methods described above.

Family	Intrusive?	Implementation cost of first experiment	Implementation cost of subsequent experiments	Hardware cost	Flexibility render time	Impact on
Traffic splitting	No	Moderate to high	Moderate to high	High	High	Low
Page rewriting	No	Moderate	Moderate	Moderate to high	Moderate	High
Client-side assignment	Yes (moderately)	Moderate	Moderate	Low	Low	High
Server-side assignment	Yes (highly)	High	Moderate to low	Low	Very high	Very low

5.3 Data path

In order to compare metrics across experiment variants, a website must first record the treatment assignments of all end users who visit the site during an experiment. Then, the website must collect raw data such as page views, clicks, revenue, render time, or customer-feedback selections. Each row of this raw data must be annotated with the identifier of the variant of each experiment that the user saw on the page request. The system must then convert this raw data into *metrics*—numerical summaries that can be compared between variants of an experiment to determine the outcome. Metrics can range from simple aggregates (total page views) all the way to complex inferred measures (customer satisfaction or search relevance). To compute metrics, the system applies basic transformations and then aggregates the observations, grouping by experiment, variant, and any other dimensions that the experimenter wishes to analyze (for example, demographics or user agent). Additional transformations may be applied at this point to produce more complex measures. From here, we create a table of metric values, broken down by dimensions, experiment, and (most importantly) variant. We can now compare metric values between variants and determine statistical significance using either any of a number of statistical tests.

Although the basic analysis techniques closely resemble those used in online analytic processing (OLAP), website experimentation raises some specific data issues.

5.3.1 Event-triggered filtering

Data collected from web traffic on a large site typically has tremendous variability, thereby making it difficult to run an experiment with sufficient power to detect effects on smaller features. One critical way to control this variability is to restrict the analysis to only those users who were impacted by the experiment (see Sect. 3.2.3). We can further restrict the analysis to the portion of user behavior that was affected by the experiment. We refer to these data restrictions as *event-triggered filtering*.

Event-triggered filtering is implemented by tracking the time at which each user first saw content that was affected by the experiment. This data can be collected directly (by recording an event when a user sees experimental content) or indirectly (by identifying experimental content from page views or other parts of the existing raw data stream). It is also possible to integrate event-triggered filtering directly into the assignment method.

5.3.2 Raw data collection

Collecting the raw observations is similar to basic website instrumentation. However, the needs of experimentation make some options more attractive than others.

5.3.2.1 Using existing (external) data collection Many websites already have some data collection in place, either through an in-house system or an external metrics provider like Omniture or Webmetrics. For these websites, a simple approach is to push the treatment assignment for each user into this system so that it becomes available for analysis. While this approach is simple to set up, most existing data collection systems are not designed for the statistical analyses that are required to correctly analyze the results of a controlled experiment. Therefore, analysis requires manual extraction of the data from the external system, which can be expensive and also precludes real-time analysis. Moreover, the existing code needs to be modified each time a new experiment is run to add the treatment assignment to all of the recorded observations. We recommend this approach only in situations where no other approach can be used.

5.3.2.2 Local data collection Using this method, the website records data locally, either through a local database or log files. The data is collected locally on each server in the fleet and must be sorted and aggregated before analysis can begin. This method can be made to scale up to very large websites. However, as the fleet scales up, collecting these logs in near real-time while minimizing data loss becomes extremely difficult. Moreover, this method makes it difficult to collect data from sources other than the webserver (like backend services or even the user's browser via JavaScript); every additional source of data increases the complexity of the log gathering infrastructure.

5.3.2.3 Service-based collection Under this model, the website implements a service specifically designed to record and store observation data. Service calls may be placed in a number of locations, including web servers, application servers, backend algorithm services, and even the end user's browser (called via JavaScript). Implementations of this model typically cache some data locally to avoid making an excessive number of physical service calls. This approach has the advantage of centralizing all observation data, making it available for easy analysis. In particular, it makes it easy to combine observations from backend services with client-side JavaScript data collection that is necessary to accurate capture user behavior on pages making extensive use of DHTML and Ajax. This method also makes it easier to experiment on large websites built on heterogeneous architectures.

Unlike with assignment methods, there is a clear winner among data collection techniques: service-based collection is the most flexible and therefore preferred when possible.

6 Lessons learned

The difference between theory and practice is larger in practice than the difference between theory and practice in theory

– Jan L.A. van de Snepscheut

Many theoretical techniques seem well suited for practical use and yet require significant ingenuity to apply them to messy real world environments. Controlled experiments are no exception. Having run a large number of online experiments, we now share several practical lessons in three areas: (i) analysis; (ii) trust and execution; and (iii) culture and business.

6.1 Analysis

The road to hell is paved with good intentions and littered with sloppy analysis

– Anonymous

6.1.1 Mine the data

A controlled experiment provides more than just a single bit of information about whether the difference in OECs is statistically significant. Rich data is typically collected that can be analyzed using machine learning and data mining techniques. For example, an experiment showed no significant difference overall, but a population of users with a specific browser version was significantly worse for the Treatment. The specific Treatment feature, which involved JavaScript, was buggy for that browser version and users abandoned. Excluding the population from the analysis showed positive results, and once the bug was fixed, the feature was indeed retested and was positive.

6.1.2 Speed matters

A Treatment might provide a worse user experience because of its performance. Linden (2006b, p. 15), wrote that experiments at Amazon showed a 1% sales decrease for an additional 100msec, and that a specific experiment at Google, which increased the time to display search results by 500 msecs reduced revenues by 20% (based on a talk by Marissa Mayer at Web 2.0). Recent experiments at Microsoft Live Search (Kohavi 2007, p. 12) showed that when the search results page was slowed down by one second, queries per user declined by 1% and ad clicks per user declined by 1.5%; when the search results page was slowed down by two seconds, these numbers more than doubled to 2.5% and 4.4%.

If time is not directly part of your OEC, make sure that a new feature that is losing is not losing because it is slower.

6.1.3 Test one factor at a time (or not)

Several authors (Peterson 2004, p. 76; Eisenberg 2005) recommend testing one factor at a time. We believe the advice, interpreted narrowly, is too restrictive and can lead organizations to focus on small incremental improvements. Conversely, some companies are touting their fractional factorial designs and Taguchi methods, thus introducing complexity where it may not be needed. While it is clear that factorial designs allow for joint optimization of factors, and are therefore superior in theory (Mason et al. 1989; Box et al. 2005) our experience from running experiments in online web sites is that interactions are less frequent than people assume (van Belle 2002), and awareness of the issue is enough that parallel interacting experiments are avoided. Our recommendations are therefore:

- Conduct single-factor experiments for gaining insights and when you make incremental changes that could be decoupled.
- Try some bold bets and very different designs. For example, let two designers come up with two very different designs for a new feature and try them one against the other. You might then start to perturb the winning version to improve it further. For backend algorithms it is even easier to try a completely different algorithm (e.g., a new recommendation algorithm). Data mining can help isolate areas where the new algorithm is significantly better, leading to interesting insights.
- Use full or fractional factorial designs suitable for estimating interactions when several factors are suspected to interact strongly. Limit the number of values per factor and assign the same percentage to the treatments as to the control. This gives your experiment maximum power to detect effects.

6.2 Trust and execution

In God we trust, all others pay cash
– Jean Shepherd

6.2.1 Run continuous A/A tests

Run A/A tests (see Sect. 3.1) and validate the following.

1. Are users split according to the planned percentages?
2. Is the data collected matching the system of record?
3. Are the results showing non-significant results 95% of the time?

Continuously run A/A tests in parallel with other experiments.

6.2.2 Automate ramp-up and abort

As discussed in Sect. 3.3, we recommend that experimenters gradually increase the percentage of users assigned to the Treatment(s). An experimentation system that analyses the experiment data in near-real-time can automatically shut-down a Treatment if it is significantly underperforming relative to the Control. An auto-abort simply

reduces the percentage of users assigned to the underperforming Treatment to zero. Since the system will automatically reduce the risk of exposing many users to egregious errors, the organization can make bold bets and innovate faster. Ramp-up is quite easy to do in online environments, yet hard to do in offline studies. We have seen no mention of these practical ideas in the literature, yet they are extremely useful.

6.2.3 Determine the minimum sample size

Decide on the statistical power, the effect you would like to detect, and estimate the variability of the OEC through an A/A test. Based on this data you can compute the minimum sample size needed for the experiment and hence the running time for your web site. A common mistake is to run experiments that are underpowered. Consider the techniques mentioned in Sect. 3.2 point 3 to reduce the variability of the OEC. Also recognize that some metrics have poor power characteristics in that their power actually degrades as the experiment runs longer. For these metrics it is important that you get an adequate number of users into the test per day and that the Treatment and Control groups are of equal size.

6.2.4 Assign 50% of users to treatment

One common practice among novice experimenters is to run new variants for only a small percentage of users. The logic behind that decision is that in case of an error only few users will see a bad Treatment, which is why we recommend Treatment ramp-up. In order to maximize the power of an experiment and minimize the running time, we recommend that 50% of users see each of the variants in an A/B test. Assuming all factors are fixed, a good approximation for the multiplicative increase in running time for an A/B test relative to 50%/50% is $1/(4p(1-p)$ where the Treatment receives portion p of the traffic. For example, if an experiment is run at 99%/1%, then it will have to run about 25 times longer than if it ran at 50%/50%.

6.2.5 Beware of day of week effects

Even if you have a lot of users visiting the site, implying that you could run an experiment for only hours or a day, we strongly recommend running experiments for at least a week or two, then continuing by multiples of a week so that day-of-week effects can be analyzed. For many sites the users visiting on the weekend represent different segments, and analyzing them separately may lead to interesting insights. This lesson can be generalized to other time-related events, such as holidays and seasons, and to different geographies: what works in the US may not work well in France, Germany, or Japan.

Putting 6.2.3, 6.2.4, and 6.2.5 together, suppose that the power calculations imply that you need to run an A/B test for a minimum of 5 days, if the experiment were run at 50%/50%. We would then recommend running it for a week to avoid day-of-week effects and to increase the power over the minimum. However, if the experiment were run at 95%/5%, the running time would have to be increased by a factor of 5–25 days, in which case we would recommend running it for four weeks. Such an experiment

should not be run at 99%/1% because it would require over 125 days, a period we consider too long for reliable result; factors, such as cookie churn, that have secondary impact in experiments running for a few weeks may start contaminating the data.

6.3 Culture and business

> *It is difficult to get a man to understand something when his salary depends upon his not understanding it.*
>
> – Upton Sinclair

6.3.1 Agree on the OEC upfront

One of the powers of controlled experiments is that it can objectively measure the value of new features for the business. However, it best serves this purpose when the interested parties have agreed on how an experiment is to be evaluated **before** the experiment is run.

While this advice may sound obvious, it is infrequently applied because the evaluation of many online features is subject to several, often competing objectives. OECs can be combined measures, which transform multiple objectives, in the form of experimental observations, into a single metric. In formulating an OEC, an organization is forced to weigh the value of various inputs and decide their relative importance. A good technique is to assess the *lifetime value* of users and their actions. For example, a search from a new user may be worth more than an additional search from an existing user. Although a single metric is not required for running experiments, this hard up-front work can align the organization and clarify goals.

6.3.2 Beware of launching features that "do not hurt" users

When an experiment yields no statistically significant difference between variants, this may mean that there truly is no difference between the variants or that the experiment did not have sufficient power to detect the change. In the face of a "no significant difference" result, sometimes the decision is made to launch the change anyway "because it does not hurt anything." It is possible that the experiment is negative but underpowered.

6.3.3 Weigh the feature maintenance costs

An experiment may show a statistically significant difference between variants, but choosing to launch the new variant may still be unjustified because of maintenance costs. A small increase in the OEC may not outweigh the cost of maintaining the feature.

6.3.4 Change to a data-driven culture

Running a few online experiments can provide great insights into how customers are using a feature. Running frequent experiments and using experimental results as major input to company decisions and product planning can have a dramatic impact on company culture. As Mike Moran said in his wonderful book "Do it Wrong Quickly" (Moran 2007) "Sometimes you have to kiss a lot of frogs to find one prince. So how can you find your prince faster? By finding more frogs and kissing them faster and faster." Software organizations shipping classical software developed a culture where features are completely designed prior to implementation. In a web world, we can integrate customer feedback directly through prototypes and experimentation. If an organization has done the hard work to agree on an OEC and vetted an experimentation system, experimentation can provide real data and move the culture towards attaining shared goals rather than battle over opinions.

7 Summary

> *Almost any question can be answered cheaply, quickly and finally, by a test campaign. And that's the way to answer them – not by arguments around a table. Go to the court of last resort – buyers of your products.*
>
> – Claude Hopkins, Scientific Advertising, 1923

> *…the ability to experiment easily is a critical factor for Web-based applications. The online world is never static. There is a constant flow of new users, new products and new technologies. Being able to figure out quickly what works and what doesn't can mean the difference between survival and extinction.*
>
> – Hal Varian, 2007

Classical knowledge discovery and data mining provide insight, but the patterns discovered are correlational and therefore pose challenges in separating useful actionable patterns from those caused by "leaks" (Kohavi et al. 2004). Controlled experiments neutralize confounding variables by distributing them equally over all values through random assignment (Keppel et al. 1992), thus establishing a causal relationship between the changes made in the different variants and the measure(s) of interest, including the Overall Evaluation Criterion (OEC). Using data mining techniques in this setting can thus provide extremely valuable insights, such as the identification of segments that benefit from a feature introduced in a controlled experiment, leading to a virtuous cycle of improvements in features and better personalization.

The basic ideas in running controlled experiments are easy to understand, but a comprehensive overview for the web was not previously available. In addition, there are important new lessons and insights that we shared throughout the paper, including generalized architectures, ramp-up and aborts, the practical problems with randomization and hashing techniques, and organizational issues, especially as they relate to OEC.

Software features in products today are commonly determined by the same way medicine was prescribed prior to World War II: by people who were regarded as experts, not by using scientific methods, such as controlled experiments. We can do better today, especially with our access to customer behavior online. In *The Progress of Experiment: Science and Therapeutic Reform in the United States, 1900–1990* (Marks 2000, p. 3), the author wrote about the increasing importance of designed experiments in the advance of medical knowledge: "Reformers in the second half of the century abandoned their predecessors' trust in the judgment of experienced clinicians. In its place, they offered an impersonal standard of scientific integrity: the double-blind, randomized, controlled clinical trial."

Many organizations have strong managers who have strong opinions, but lack data, so we started to use the term HiPPO, which stands for Highest Paid Person's Opinion, as a way to remind everyone that success really depends on the users' perceptions. Some authors have called experimentation the "New Imperative for Innovation" (Thomke 2001) and point out that "new technologies are making it easier than ever to conduct complex experiments quickly and cheaply." We agree and believe that companies can accelerate innovation through experimentation because it is the customers' experience that ultimately matters, and we should listen to them all the time by running experiments.

Acknowledgements We would like to thank members of the Experimentation Platform team at Microsoft. The first author wishes to thank Avinash Kaushik for a great conversation on A/B testing, where he used the term "HiPO" for Highest Paid Opinion; this evolved into HiPPO (picture included) in our presentations. We thank Fritz Behr, Keith Butler, Rod Deyo, Danyel Fisher, James Hamilton, Andrew Hesky, Ronit HaNegby, Greg Linden, Foster Provost, Saharon Rosset, Geoff Webb, and Zijian Zheng for their feedback.

Appendix A

When randomization by user-ID is not appropriate

The approach we describe in this paper is to randomly assign users to one group or another and compare these groups of users to determine which experience (i.e. Treatment) is best. There are some experimentation objectives where this approach will not work. We will describe three of these and alternative approaches to randomization in an online environment.

1. *Control may affect the effectiveness of the Treatment and vice versa* Bidding on Ebay.[2] Suppose the Treatment is to give an incentive (perhaps a $5 discount or certain percent off the final bid price) for a user to be the first bidder and no such incentive exists for the Control. Assume the success metric (OEC) is the ratio of the final sales price to the minimum bid for each item. If some users have this incentive and others do not, the presence of the Treatment will affect all items so we cannot get a true measure of the effectiveness of making this change. In this case you can randomly assign one group of items in the auction to be in the Control and the rest to be in the Treatment and compare the OEC for these two groups. i.e. randomly assign the items in the auction, not the users.

[2] We wish to thank Neel Sundaresan and his team at eBay for this example.

2. *Not desirable to randomize based on user* Price elasticity study. The usual randomization based on user is not desirable because bad customer relations could result if it's exposed that some customers are getting a different price than other customers (everything else being equal) as Amazon.com discovered when it ran such a study (Weiss 2000). Here also, the items involved in the study can be randomly assigned to the Treatment or Control instead of randomizing the users.
3. *Not possible to randomize on user* Search Engine Optimization (SEO). Most robots do not set cookies so they would not be in any experiment. If you wanted to conduct a test on robot behavior (e.g. clickthroughs by robots or other) you cannot randomize based on a user ID. Instead you can take groups of pages on your site that are similar and randomly assign pages within each group to Treatment or Control and compare robot behavior for the two groups of pages.

References

Alt B, Usborne N (2005) Market Exp J. [Online] December 29, 2005. http://www.marketingexperiments.com/improving-website-conversion/multivariable-testing.html

Boos DD, Hughes-Oliver JM (2000) How large does n have to be for Z and t intervals? Am Statist 54(2):121–128

Box GEP, Hunter JS, Hunter WG (2005) Statistics for experimenters: design, innovation, and discovery, 2nd edn. Wiley, ISBN: 0471718130

Burns M (2006) Web analytics spendings trends 2007. Forrester Research Inc., Cambridge

Charles RS, Melvin MM (2004) Quasi experimentation. [book auth.] In: Wholey JS, Hatry HP, Newcomer KE (eds) Handbook of practical program evaluation, 2nd edn. Jossey-Bass

Chatham B, Temkin BD, Amato M (2004) A primer on A/B testing. Forrester Research

Davies OL, Hay WA (1950) Construction and uses of fractional factorial designs in industrial research. Biometrics 233(6):121–128

Eisenberg B (2003a) How to Decrease sales by 90%. ClickZ. [Online] Feb 21, 2003. http://www.clickz.com/showPage.html?page=1588161

Eisenberg B (2003b) How to increase conversion rate 1,000%. ClickZ. [Online] Feb 28, 2003. http://www.clickz.com/showPage.html?page=1756031

Eisenberg B (2004) A/B testing for the mathematically disinclined. ClickZ. [Online] May 7, 2004. http://www.clickz.com/showPage.html?page=3349901

Eisenberg B (2005) How to improve A/B testing. ClickZ Netw. [Online] April 29, 2005. http://www.clickz.com/showPage.html?page=3500811

Eisenberg B, Eisenberg J (2005) Call to action, secret formulas to improve online results. Wizard Academy Press, Austin, 2005. Making the dial move by testing, introducing A/B testing

Eisenberg B, Garcia A (2006) Which sells best: a quick start guide to testing for retailers. Future now's publications. [Online] 2006. http://futurenowinc.com/shop/

Forrester Research (2005) The state of retailing online. Shop.org

Google Website Optimizer (2008) [Online] 2008. http://services.google.com/websiteoptimizer

Hawthorne effect (2007) Wikipedia. [Online] 2007. http://en.wikipedia.org/wiki/Hawthorne_experiments

Hopkins C (1923) Scientific advertising. Crown Publishers Inc., New York City

Kaplan RS, Norton DP (1996) The balanced scorecard: translating strategy into action. Harvard Business School Press, ISBN: 0875846513

Kaushik A (2006) Experimentation and testing: a primer. Occam's Razor by Avinash Kaushik. [Online] May 22, 2006. http://www.kaushik.net/avinash/2006/05/experimentation-and-testing-a-primer.html

Keppel G, Saufley WH, Tokunaga H (1992) Introduction to design and analysis, 2nd edn. W.H. Freeman and Company

Kohavi R (2007) Emetrics 2007 practical guide to controlled experiments on the web. [Online] October 16, 2007. http://exp-platform.com/Documents/2007-10EmetricsExperimenation.pdf

Kohavi R, Parekh R (2003) Ten supplementary analyses to improve e-commerce web sites. WebKDD

Kohavi R, Round M (2004) In: Sterne J (ed) Front line internet analytics at Amazon.com. Santa Barbara, CA. http://ai.stanford.edu/~ronnyk/emetricsAmazon.pdf
Kohavi R et al (2004) Lessons and challenges from mining retail e-commerce data. Machine Learn 57(1–2):83–113. http://ai.stanford.edu/~ronnyk/lessonsInDM.pdf
Koselka R (1996) The new mantra: MVT. Forbes. March 11, 1996, pp 114–118
Linden G (2006a) Early Amazon: shopping cart recommendations. Geeking with Greg. [Online] April 25, 2006. http://glinden.blogspot.com/2006/04/early-amazon-shopping-cart.html
Linden G (2006b) Make data useful. [Online] Dec 2006. http://home.blarg.net/~glinden/StanfordDataMining.2006-11-29.ppt
Manning H, Dorsey M, Carney CL (2006) Don't rationalize bad site design. Forrester Research, Cambridge
Marks HM (2000) The progress of experiment: science and therapeutic reform in the united states, 1900–1990. Cambridge University Press, ISBN: 978-0521785617
Maron O, Moore AW (1994) Hoeffding races: accelerating model selection search for classification and function approximation. http://citeseer.ist.psu.edu/maron94hoeffding.html
Mason RL, Gunst RF, Hess JL (1989) Statistical design and analysis of experiments with applications to engineering and science. Wiley, ISBN: 047185364X
McGlaughlin F et al (2006) The power of small changes tested. Market Exp J. [Online] March 21, 2006. http://www.marketingexperiments.com/improving-website-conversion/power-small-change.html
Miller S (2006) The ConversionLab.com: how to experiment your way to increased web sales using split testing and Taguchi optimization. http://www.conversionlab.com/
Miller S (2007) How to design a split test. Web marketing today, conversion/testing. [Online] Jan 18, 2007. http://www.wilsonweb.com/conversion/
Moran M (2007) Do it wrong quickly: how the web changes the old marketing rules. IBM Press, ISBN: 0132255960
Nielsen J (2005) Putting A/B testing in its place. Useit.com Alertbox. [Online] Aug 15, 2005. http://www.useit.com/alertbox/20050815.html
Omniture (2008) [Online] 2008. www.omniture.com/products/optimization/offermatica
Optimost (2008) [Online] 2008. http://www.optimost.com
Peterson ET (2004) Web analytics demystified: a marketer's guide to understanding how your web site affects your business. Celilo Group Media and CafePress, ISBN: 0974358428
Peterson ET (2005) Web site measurement hacks. O'Reilly Media, ISBN: 0596009887
Plackett RL, Burman JP (1946) The design of optimum multifactorial experiments. Biometrika 33:305–325
Quarto-vonTivadar J (2006) AB testing: too little, too soon. Future Now. [Online] 2006. http://www.futurenowinc.com/abtesting.pdf
Rossi PH, Lipsey MW, Freeman HE (2003) Evaluation: a systematic approach, 7th edn. Sage Publications, Inc., ISBN: 0-7619-0894-3
Roy RK (2001) Design of experiments using the taguchi approach: 16 steps to product and process improvement. Wiley, ISBN: 0-471-36101-1
SiteSpect (2008) [Online] 2008. http://www.sitespect.com
Spool JM (2004) The cost of frustration. WebProNews. [Online] September 20, 2004. http://www.webpronews.com/topnews/2004/09/20/the-cost-of-frustration
Sterne J (2002) Web metrics: proven methods for measuring web site success. Wiley, ISBN: 0-471-22072-8
Tan P-N, Kumar V (2002) Discovery of web robot sessions based on their navigational patterns. Data Min Knowl Dis
Thomke S (2001) Enlightened experimentation: the new imperative for innovation, Feb 2001
Thomke SH (2003) Experimentation matters: unlocking the potential of new technologies for innovation
Tyler ME, Ledford J (2006) Google analytics. Wiley, ISBN: 0470053852
Ulwick A (2005) What customers want: using outcome-driven innovation to create breakthrough products and services. McGraw-Hill, ISBN: 0071408673
Usborne N (2005) Design choices can cripple a website. A list apart. [Online] Nov 8, 2005. http://alistapart.com/articles/designcancripple
van Belle G (2002) Statistical rules of thumb. Wiley, ISBN: 0471402273
Varian HR (2007) Kaizen, that continuous improvement strategy, finds its ideal environment. New York Times. February 8, 2007. Online at http://www.nytimes.com/2007/02/08/business/08scene.html?fta=y
Verster (2008) [Online] 2008. http://www.vertster.com

Weiss CH (1997) Evaluation: methods for studying programs and policies, 2nd edn. Prentice Hall, ISBN: 0-13-309725-0

Weiss TR (2000) Amazon apologizes for price-testing program that angered customers. www.Safecount.net. [Online] September 28, 2000. http://www.infoworld.com/articles/hn/xml/00/09/28/000928hnamazondvd.html

Wheeler RE (1974) Portable power. Technometrics 16:193–201. http://www.bobwheeler.com/stat/Papers/PortablePower.PDF

Wheeler RE (1975) The validity of portable power. Technometrics 17(2):177–179

Widemile (2008) [Online] 2008. http://www.widemile.com

Wikepedia (2008) Multi-armed bandit. Wikipedia. [Online] 2008. http://en.wikipedia.org/wiki/Multi-armed_bandit

Willan AR, Briggs AH (2006) Statistical analysis of cost-effectiveness data (statistics in practice). Wiley