Optimizely Stats Engine

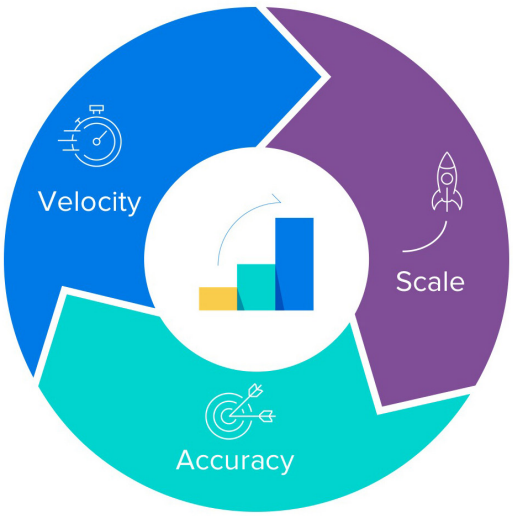
FASTER DECISIONS YOU CAN TRUST FOR DIGITAL EXPERIMENTATION

Table of Contents

Summary

Today's leading digital businesses are dominating their markets by taking advantage of experimentation to constantly improve their digital products, online commerce, and marketing campaigns. In a recent HBR[1] article, Stefan Thomke, Professor of Business Administration at Harvard Business School, talks about the astonishing power of online experimentation and why it will be essential in order to gain a competitive advantage. It's not enough to simply rely on hunches when you launch new products and campaigns. You need to replace digital guesswork with evidence-based optimization by testing your decisions and assessing the feedback based on evidence and facts.

At the core of Optimizely's experimentation platform is our Stats Engine. Optimizely Stats Engine is designed to help you make decisions you can trust, faster than ever before. Stats Engine is a unique approach that was developed in coordination with leaders in statistical research at Stanford University. Stats Engine embeds several innovations to deliver what every digital experience optimization effort needs—accelerated speed and greater scale of experimentation while also improving the accuracy of results and outcomes.



Stats Engine is designed for the digital age

1. The Surprising Power of Online Experiments, Ron Kohavi and Stefan Thomke, Harvard Business Review, September-October 2017, https://hbr.org/2017/09/the-surprising-power-of-online-experiments

Going all the way back to the t-test developed at Guinness Brewing Company in 1908, there has been a rich tradition of novel and useful statistical techniques developed specifically to answer the question of whether one sample is different from another. At Optimizely, we carry this tradition on into the digital age with Stats Engine. Simply put, Stats Engine is a modern statistical solution adapted to the unique needs of the digital experimenter.

At the heart of Stats Engine lies two, core statistical techniques. First, sequential testing enables you to monitor your data continuously and move past the limitations of traditional fixed-horizon testing to achieve significance faster. Second, built-in false discovery rate control gives you statistical guard rails to ensure consistent interpretation of your results, no matter how many variations or metrics you add to an experiment.

	Traditional Statistics	Stats Engine
Velocity	Fixed horizon approach requires waiting before results can be viewed Involves guesswork that can slow experiments	Sequential Testing lets you monitor results in real-time and stop early No guesswork required
Accuracy	Running multiple comparisons simultaneously inflates errors	Built-in False Discovery Rate (FDR) Control lets you test multiple variations and metrics without inflating errors
Scale	Forces you to choose between speed and accuracy Requires data science resources	Gives you both speed and accuracy, no need to compromise Built-in protections that eliminate risk and let everyone experiment

This whitepaper will dive deeper into what Stats Engine is, why we developed it, and how it can help you with your experimentation program.

What is a good test?

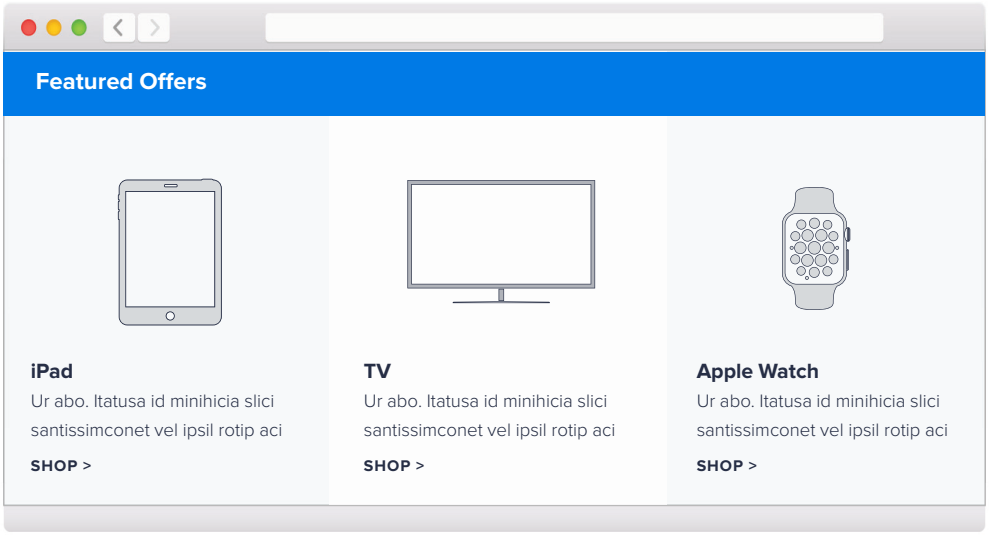
A false positive, or type-1 error, occurs if a test declares a significant difference when, in actuality, there is none. Some degree of type-1 error is inevitable in any statistical test as a consequence of dealing with randomness. In a correctly designed test, however, the false positive rate is controlled by the significance level set by the user.

At the end of the day, the success or failure of your experimentation pipeline relies on an effective way of turning noisy data into the right decision. So what constitutes a good test to achieve that? A good test should be sensitive enough to tease out the true effects hiding within the noise and be robust enough to ignore the spurious patterns emerging from the same noise, all while making efficient use of the available data to deliver results quickly.

In other words, a good test offers good **control of false positives** and **high statistical power.** These concepts have clear implications for business risks and costs.

If you are an e-commerce company, you might want to test how a change in your website's user experience is affecting conversion rates at different stages of your purchasing funnel. For example, imagine you are testing a different method to promote product offers on your site's homepage.

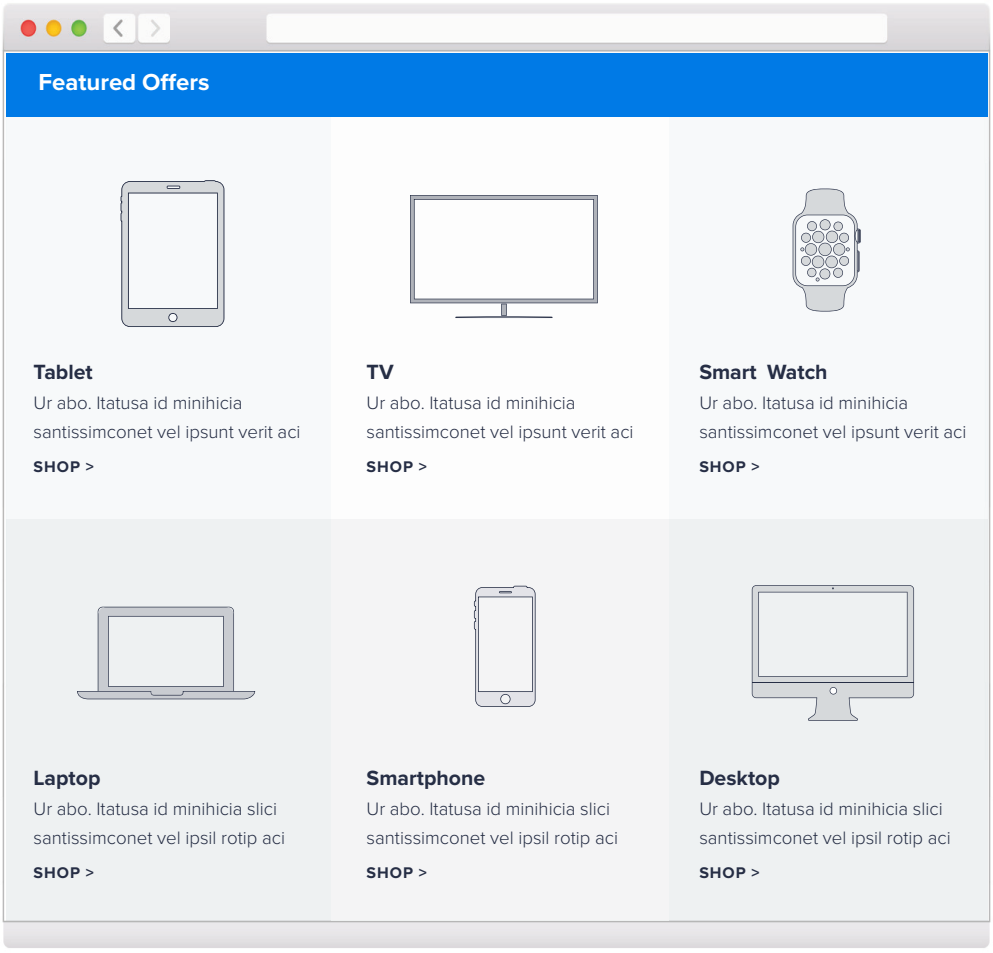
Currently, offers are shown for three products on a single row in a 1x3 arrangement.



Original: 3 product offers arranged in one row

***Statistical power** represents the ability of a test to detect a true difference when it exists. That power is quantified by type-2 errors, or instances where a test fails to declare a significant difference when one, in fact, exists. Power is defined as one minus the rate of type-2 errors.*

You are also interested in whether displaying offers in a 2x3 arrangement will have a positive impact on total cart value.

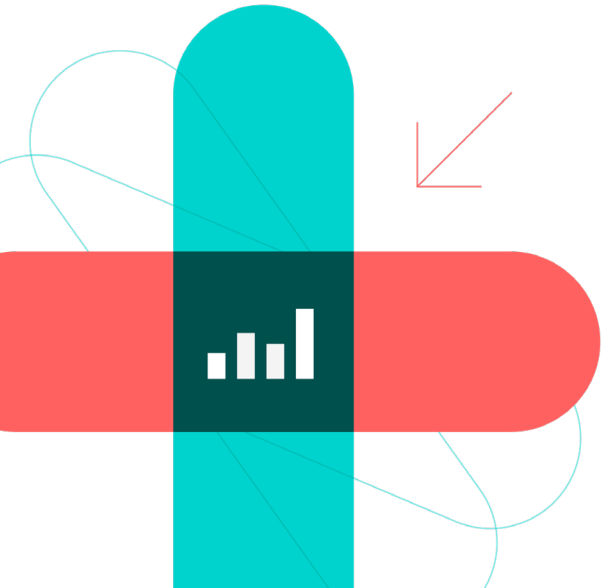


Variation 1: 2 row design with 3 product offers in each row

- A false positive in this experiment would mislead you to think that the 2x3 arrangement increases total cart value when, in fact, it does not. At best, implementing the new arrangement would waste your engineers' time and effort. At worst, it would induce a huge opportunity cost by distracting you from variations with true lifts in revenue. In addition, such a faulty arrangement could have a negative impact, with visitors adding fewer offers to their carts, resulting in millions of dollars in lost revenue.

- Now suppose that the 2x3 arrangement increases total cart value. A test with low power would often fail to declare the 2x3 arrangement a winner. In this case, the risk of missing a lift shouldn't be overlooked, because a lift of even just 1% can translate into millions of dollars.

Balancing good control of false positives with high power is the very crux of the A/B testing problem. Good statistical power can generally be achieved by setting a large required sample size. However, this increases the temptation to peek at interim results while a test is running, which increases the false positive rate. In the next two sections, we'll explain why traditional fixed-horizon tests are handcuffed by this trade-off and how Stats Engine's sequential test helps you overcome the trade-off.



Shortcomings of the traditional fixed-horizon test

THE PROPER WAY TO DO A T-TEST

The classical testing situation involves fixed time horizons. Put yourself in the shoes of William Sealy Gosset, the inventor of the t-test while at Guinness Brewing Company in 1908. In order to optimize Guinness Brewing's profit margin, one of your tasks might be to determine whether a specific strain of barley yields more sugar than another. The data collection process for such a question would have been:

1. Plant several fields of the control and treatment barley strains
2. Wait several months for the barley to grow
3. Harvest the fields and compare their respective sugar yields

In this problem, there is a long waiting period for the data to arrive, and there is no way to inspect the results in the interim. This dilemma is common to many traditional statistical applications, such as drug efficacy studies in clinical trials. In light of this, most traditional statistical tests, such as the t-test, assume a fixed sample size. However, such fixed-horizon tests are of limited usefulness in digital experimentation for these reasons:

- Today, data is available almost instantaneously. We no longer need to wait for data to be collected before calculating results. But continuous monitoring of fixed-horizon tests like the t-test dramatically inflates the type-1 error rate due to the peeking problem.

- Setting a sample size in advance requires inputting the expected lift from the experiment. Guess too low, and you will have to wait too long to detect your effect. Guess too high, and you'll have low statistical power.

- With the wealth of data available today, running many experiments at a time is now the norm. But running multiple tests simultaneously, without a correction, also inflates type-1 errors due to the multiple comparisons problem.

Let's explore some of these points in more depth.

THE PEEKING PROBLEM

Continuous monitoring of fixed-horizon tests generally results in more false positives. This phenomenon is triggered whenever an experimenter either stops a test the first moment he sees significance or continues an inconclusive test past the prescribed sample size in order to try to obtain significance.

Suppose you run a test with a significance level α. For the t-test, the procedure assumes the user only views the data once. In that case, the p-value threshold for rejection is precisely set to account for randomness appearing in that one view of the data. In other words, the test is calibrated to guarantee that you won't see a false positive more than α proportion of the time.

If a user views the data again at a later time, then the user is giving himself two chances to surface a false positive, each with probability α. The more views you allow yourself to take, the more this effect compounds, and the higher your overall chance of experiencing a false positive. It doesn't take many views to push the false positive rate sky-high.

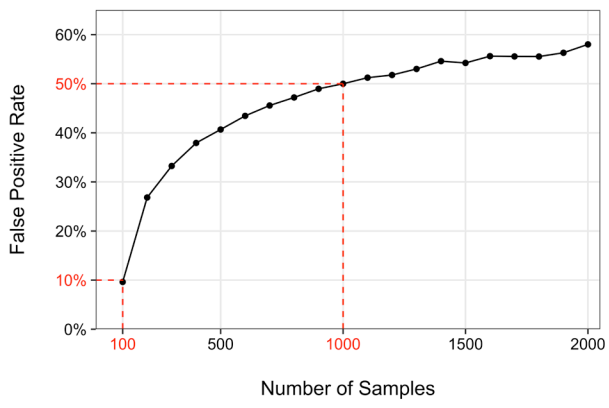


Figure 3: Observed probability of obtaining a false positive when continuously monitoring a fixed-horizon A/A test with significance level 10%, starting after 100 samples have been observed. By 1000 observations, the chance of a false positive is 50%, a 5-fold increase.

The takeaway here is that fixed-horizon tests are difficult to adapt to modern situations with data arriving in real time. In traditional agricultural settings, the end of the growing season was a natural fixed-horizon to the experimental problem, in which case a fixed-horizon technique was a perfect fit. In the digital realm, this is no longer true.

THE GUESSING PROBLEM

The other drawback to the fixed-horizon framework is the sample size calculation. The sample size crucially depends on one particular input from the user—the test's desired minimum detectable effect (MDE). However, a potential mismatch between the MDE and the unknown, true effect due to the variation can have dire consequences.

- When the MDE is lower than the real effect size, the calculated sample size will be higher than necessary, and you'll experience long wait times for the test to conclude.

- When the MDE is higher than the real effect size, the test will be underpowered with a high chance that the test will fail to report any significance at all.

Therefore, good performance in fixed-horizon tests pivots on a user's ability to accurately estimate the MDE. As with the peeking problem, this is another point where fixed-horizon tests actually fit reasonably well with traditional applications. If the sample is being handed to you, or you have a lot of a priori knowledge on what effect sizes to expect, then this calculation is not a problem. If, however, you are a digital experimenter moving rapidly, you might face some consequences from inaccurate estimates on how much data to use.

Stats Engine's sequential test

At Optimizely, we take a sequential testing approach that avoids the peeking and guessing problems entirely. The benefits of sequential testing are clear:

1. **Continuous monitoring is fundamental to sequential testing.**
Stats Engine was specifically designed to let you monitor results in real-time so that you can make decisions more quickly, without sacrificing the integrity of your data.

2. **The sequential test achieves optimal statistical power.**
The statistical power of a sequential test naturally increases as the test runs, so detecting small effects no longer requires any arbitrary guesses about your effect sizes.

3. **Continuous monitoring plus optimal power means faster results.**
Stats Engine adapts to the true effect size automatically and allows you to stop early for larger-than-expected effect sizes, enabling faster times to significance on average.

How fast? We found that if the lift of your A/B test is 5 percentage points higher than your MDE, Stats Engine will run about as fast as an ordinary t-test will. As soon as the improvement exceeds the MDE by as much as 7.5 percentage points, Stats Engine is closer to 75% faster. For larger experiments (>50,000 visitors), the gains are even higher, and Stats Engine can call a winner or loser up to 2.5 times as fast.

Traditional Statistics

Fixed-horizon approach requires waiting before results can be viewed

Invoves guesswork that can slow experiments

Optimizely Stats Engine

Sequential Testing lets you monitor results in real-time and stop early

No guesswork required

Make faster decisions by monitoring results in real-time

So what goes on underneath the hood? There are two key concepts to understanding the sequential nature of Stats Engine: the likelihood ratio and always valid p-values.

THE LIKELIHOOD RATIO

Most fixed-horizon tests make decisions based on the scaled difference between the sample means of the control and treatment. The sequential test uses a statistic called the mixture likelihood ratio.

$$\tilde{\Lambda}_n = \int_{\Theta} \Lambda_n(\theta)\, dH(\theta)$$

$$\begin{aligned} \Lambda_n(\theta) &= \text{relative likelihood that the true effect size is } \theta \text{ compared to } 0 \\ &\quad\ \text{using data observed up until time } n \\ H(\theta) &= \text{prior distribution on the set of possible true effect sizes } \Theta \end{aligned}$$

Equation 1: The formula for the mixture likelihood ratio after n observations.

In brief, this ratio is a smart summary of the observed data at time n which quantifies the evidence in favor of a true lift of θ compared to 0, averaged over all possible true lifts weighted by a prior distribution on the space of possible true lifts Θ. Large values of this ratio represent evidence in favor of the treatment being better than the control, while low values represents evidence in favor of there being no effect.

The specific prescription we employ is to declare a winner (or loser) if this ratio ever rises above the (predetermined) threshold $1/\alpha$. This is the precise number that guarantees false positive rate control at no more than α.

The importance of the specific form of this statistic is that it enables the test to run sequentially and adapt to the unknown true lift of the variation as more and more data comes in. This guarantees that the test is able to detect even very small differences between the control and variation without a sample size calculation up front.

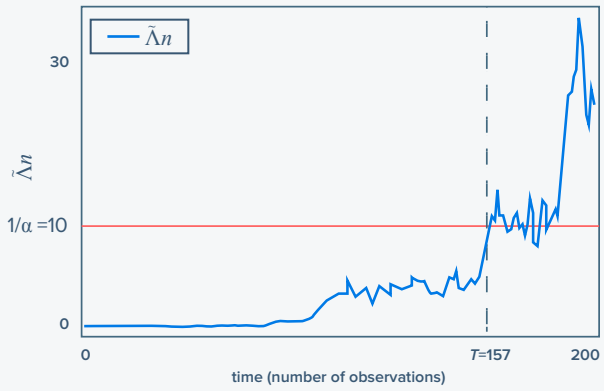


Figure 4: A sequential test's view of an A/B test with 200 observations. The blue line tracks the mixture likelihood ratio over time.

Setting a rejection threshold at 1/0.10 guarantees that the sequential test will report a false positive 10% of the time at most. After 157 samples, the mixture likelihood ratio rises past this threshold, indicating sufficient evidence to declare the variation a winner at that point.

FROM THE SEQUENTIAL TEST TO ALWAYS-VALID P-VALUES

The recipe above defines the mechanism by which Stats Engine calls winners and losers. But, what if you need to quantify the strength of evidence as the test progresses, before you have enough evidence to declare a winner or loser?

With Stats Engine, you have access to always-valid p-values and confidence intervals. These are results which can be interpreted at any point during the running experiment without increasing the risk for errors. They provide you with a powerful tool for assessing the strength of a signal during a running experiment. At Optimizely, we adhere to the convention of representing p-values as "significance" by the formula (1 - p-value) X 100.

How do we construct these p-values from a sequential test? In general, a p-value is a measure of how inconsistent the observed data are with an A/A test. In a sequential test, the p-value is the smallest significance level such that it would still allow us to call a winner (or loser) using the currently-observed data.

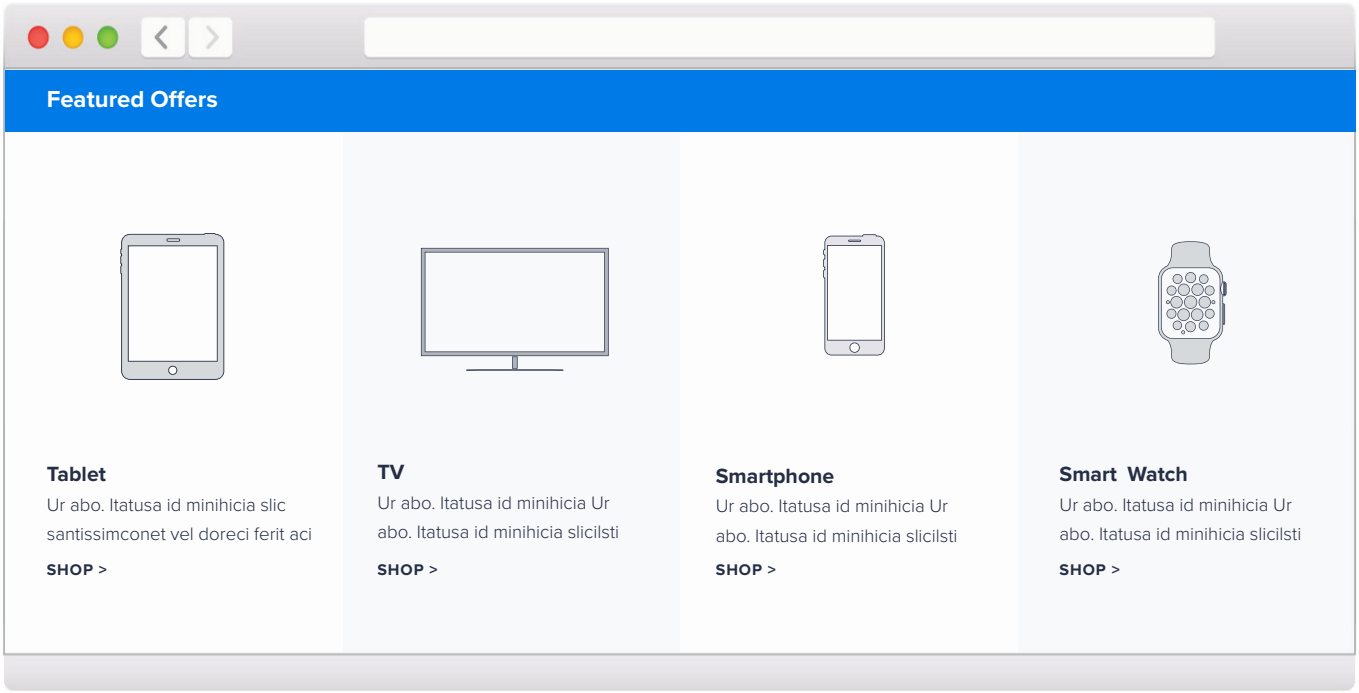
We defer a full discussion of always-valid p-values to the technical paper[2], but emphasize that using always-valid p-values (significance) to call winners and losers (by comparing it to the predefined significance level) gives us exactly the false positive rate control and power characteristics that we want, while enabling interpretability of strength of evidence at any point during the experiment.

2. The New Stats Engine, Leo Pekelis, David Walsh, and Ramesh Johari. http://pages.optimizely.com/rs/optimizely/images/stats_engine_technical_paper.pdf

Stats Engine's false discovery rate control

Recall, for a moment, our e-commerce example. Now suppose, that instead of testing a simple 2x3 layout versus the original 1x3 layout, you're interested in a testing both a 1x4 layout and a 2x3 layout. In addition, you would like to know the impact of these layouts on customer visit frequency and average order value, as well as on total cart value.

You are now looking at 2 variations along 3 different metrics. In total, you are making 6 comparisons. Unfortunately, using traditional statistics in a multiple comparisons situation like this will result in a large increase in false positives. That means the chance of at least one variation being declared a winner incorrectly is much higher than expected.



Variation 2: 4 product offers in one row

Stats Engine comes with false discovery rate control that lets you test multiple variations and metrics without inflating errors. A correction for false discoveries is automatically enabled for all experiments. Let's explore how this works in more depth.

THE MULTIPLE COMPARISONS PROBLEM

False discovery rate control solves the multiple comparisons problem, the phenomenon of increased false positives when the number of variations or metrics increases.

The problem is easy to understand. Every individual test has some chance of reporting a false positive. Therefore, if you simply perform enough tests simultaneously, you are eventually bound to see at least one false positive within that group of tests. The multiple comparisons problem is similar to the peeking problem in that neither are inherent flaws of statistical tests, but rather artifacts triggered by use cases commonly encountered in digital experimentation. In the case of peeking, the use case is continuous monitoring of streaming data. In the case of multiple comparisons, the use case is parallel experimentation of multiple variations and metrics.

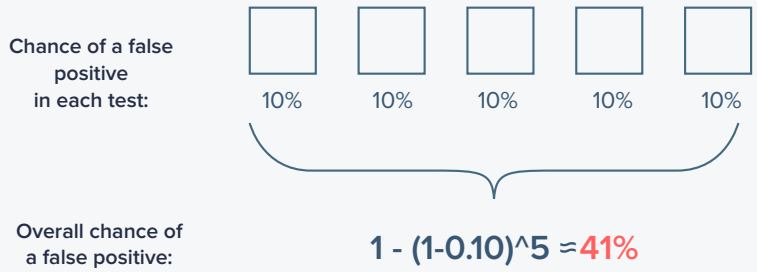


Figure 5: The chance of at least one false positive in a group of 5 independent tests.

Even with 5 comparisons at a reasonable significance level of 10%, the chance of at least one false positive is more than quadruple the chance of a false positive in any individual test when the tests are independent.

At Optimizely, each new variation added to a test (or new metric being tracked) is considered a separate test. Therefore, the experimenter wishing to test several variations at a time is faced with the task of balancing the increase in potential winners with the increased risk that each potential winner is only due to random chance.

WHY FALSE DISCOVERY RATE CONTROL IS A BETTER APPROACH

The false positive rate is important to control, but we can do even better. At Optimizely, we control the false discovery rate (FDR), a concept related to the false positive rate but easier to interpret and more relevant to decision-making in a business context.

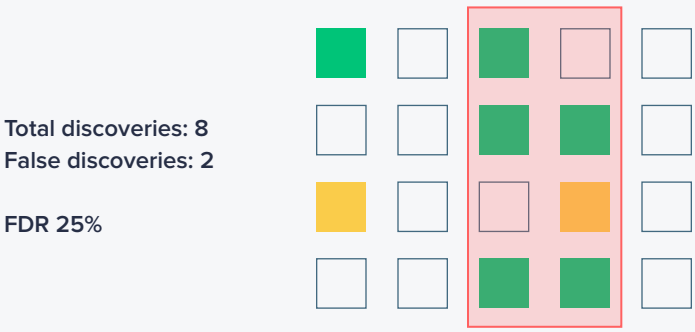


Figure 6: Chance of a false positive in each test
Green/yellow boxes indicate declared winners/losers. If the red outline denotes tests that contain true winners and losers, then the false discovery rate is the proportion of green and yellow boxes which fall outside of the red outline out of the total number of green and yellow boxes.

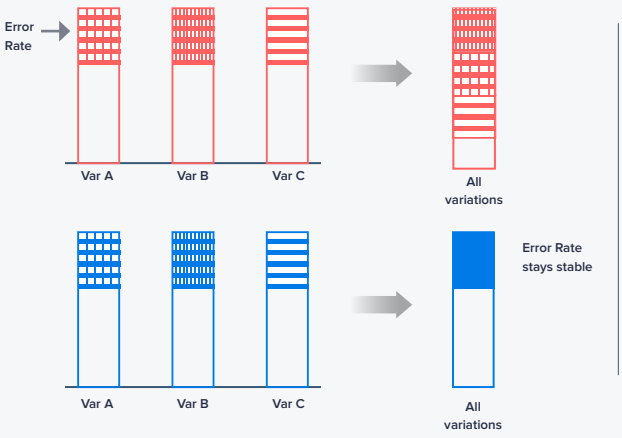
To see why FDR is generally more actionable, consider how the decision maker might react to seeing 5 conclusive tests.

If all that's known is the decision procedure controlled type-1 error, then the number of other tests run is very important. If 50 other tests were run, it's highly likely these 5 may be conclusive by random chance alone. However, if only 5 other tests were run, then the decision maker may become more confident the 5 conclusive results are meaningful. In other words, the number of other inconclusive tests affects whether the decision maker believes his results are actionable.

This perverse consequence is eliminated when using procedures that control FDR. Reporting a set of A/B tests with, for example, "false discovery rate less than 10%" is completely actionable without knowing anything about non-significant variations. The statement means that at least 90% of the reported red and green boxes have a true difference between variation and baseline, regardless of how many remaining boxes are in the matrix. By themselves, they are a very promising group of variations to act on in order to optimize your business.

OPTIMIZELY'S FDR CORRECTING PROCEDURE

Optimizely employs the Benjamini-Hochberg (BH) procedure for controlling FDR. Our version of the method automatically corrects significance values so that, by using the deflated values to determine when to call winners and losers, the FDR is properly controlled. For example, in an experiment with significance level set at 90%, Stats Engine FDR correction will produce corrected significance values that, when compared to the 90% threshold to declare winners, will always guarantee FDR of no greater than 90% in that experiment.



Achieve high accuracy using FDR Control

The BH procedure applies a different level of deflation to each metric-variation according to its rank by significance. The central idea is that the relative ranked significance values gives you information about the relative prevalence of true positives, and therefore enables good control of the FDR as opposed to the false positive rate.

Displaying these automatically-corrected values frees the user from the BH procedure's complicated definition. It also allows direct interpretation of the significance value in the context of a correction. As a result, no extra steps are needed to rigorously control for multiple comparisons. For this reason, Optimizely calculates significance as (1 - FDR) so that significance now represents the chance that a significant test is a true winner or loser, rather than the chance that the significant result is actually a fluke arising from an A/A test.

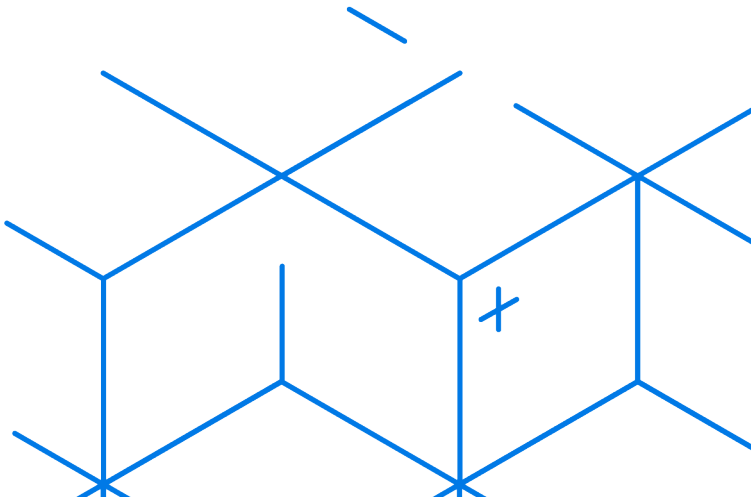
Experiment at scale across your organization

Statistics is the science of data. Its long history has produced an extensive menu of techniques that can be adapted to a wide array of modern, data-driven problems. This is both a blessing and a curse—a blessing because of the wealth of choices with which to tackle any problem, and a curse because of the possibility of choosing the wrong one.

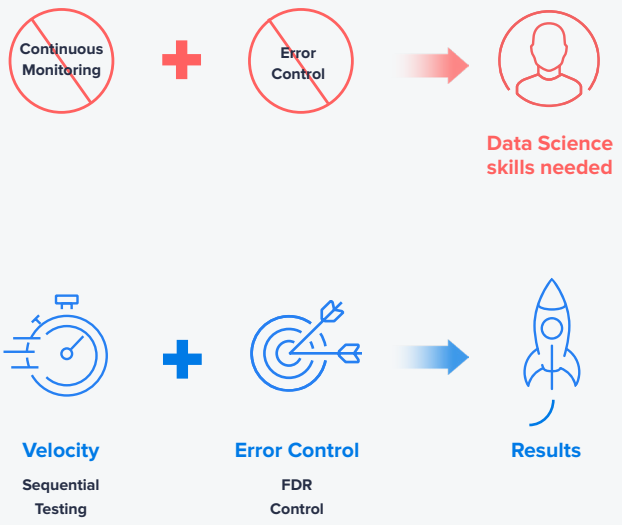
At Optimizely, we bring the right tools to the job. Stats Engine allows you to experiment with velocity and accuracy and, most importantly, scale your entire program. The more experiments you run, the higher the chances of those experiments making a positive impact on your business.

Traditional statistics forces you to choose between optimizing for speed or for accuracy. If you choose speed, you increase your error rate. If you choose accuracy, experiments take too long. Alternatives require you to have advanced data science skills in order to control for errors in the data and to interpret results accurately.

With Optimizely, you don't have to make that compromise and can therefore scale experimentation to constantly improve your digital products and experiences. Stats Engine automatically manages speed and accuracy for you.



Today, we have the tools to collect an enormous quantity of data quickly and inspect it as it arrives. Sequential testing leverages that capability to accelerate your experiments while reducing the guesswork involved. Also, false discovery rate control gives your teams statistical guard-rails and eliminates risk, so they can then experiment at a volume commensurate with your data. Because we remove so much work from your analytics team, Optimizely can enable your other teams in the organization to experiment and make decisions based on the results.



Traditional Statistics

Forces you to choose between speed and accuracy

Requires data science resources

Optimizely Stats Engine

Gives you both speed and accuracy, no need to compromise

Built-in protections that eliminate risk and let everyone experiment

Glossary

Effect Size
The amount of difference between the treatment and control of a test.

False Positive Rate
Computed by dividing the number of false positives by (total number of false positives + true negatives.)

False Discovery Rate
The expected number of false discoveries—incorrect winners and losers—computed by dividing the number of false positives by the total number of significant results.

P-value
A value that generally answers the question: if two variations are indeed the same, how likely is it that the observed conversion rate difference (improvement) was due to random chance?

Statistical Power
Sometimes expressed as (1 - the type II error rate), this is the probability that an experimenter will detect a difference when it exists. It is also the probability of correctly rejecting a null hypothesis. In Stats Engine, all experiments are adequately powered.

Statistical Significance
The number in an experiment results report that represents the likelihood that the difference between a variation and the original (the effect) is not due to chance. In Stats Engine, this is displayed as a value between 0 and 99%, or (1 - the false discovery rate).

Get Started Today

Join global digital leaders and make experimentation an integral part of building and marketing your digital products and experiences. Learn how Optimizely can help transform your business.

Visit **Optimizely.com** today to learn more.