The Open Guide to Successful AB Testing

with a focus on GrowthBook

version 1.0

GrowthBook

Contents

Introduction

In today's data-driven world, businesses of all sizes rely on A/B testing to make data-driven decisions. A/B testing, also known as split testing, has come a long way from a simple tool to optimize websites, and is often used as a powerful tool to determine the impact of any changes to your application.
By measuring how your user behavior and engagement changes in a controlled manner, you can determine causally if your hypothesis is correct, and make informed data-driven decisions that improve user experience, increase conversions, and drive growth.

This document is intended to be an open source and continuously updated guide to A/B testing. Whether you're a seasoned expert at running experiments, or just starting out, this book will provide you with the knowledge and skills you need to run a successful A/B testing program, with a specific focus on GrowthBook, an open source feature flagging and A/B testing platform.

In the following chapters, we'll start with an overview of what A/B testing is, and familiarize you with the terms that are commonly used. We'll cover the basics of statistical significance, sample size, and other key concepts that are essential for understanding A/B testing.

Next, we'll cover the best practices for running an A/B test, followed by some of the common mistakes and pitfalls that can affect experiment programs.

Finally, we'll go beyond individual A/B tests and talk about how to run an experimentation program, and then specifics of how to do this well with GrowthBook. By the end of this book, you'll have a deep understanding of A/B testing and be ready to apply this powerful technique to your own projects, hopefully using GrowthBook.

A/B Testing Fundamentals

If you are new to A/B testing, you may find a lot of new terminology. The goal of this section is to help you understand the basics of A/B testing.

Terms

Control (or Baseline)

The existing version of the product that you are trying to improve upon.

Variation (or Treatment)

A new version of the page that you are testing against the Control.

Hypothesis

Formal way to describe what you are changing and what you think it will do.

Statistical Significance

an indicator that the difference in performance between the control and treatment groups is unlikely to have occurred by chance.

Confidence level

The level of certainty we want before the result of a test is statistically significant. A common confidence level used in A/B testing is 95%.

Sample size

The number of visitors or users who are included in the A/B test.

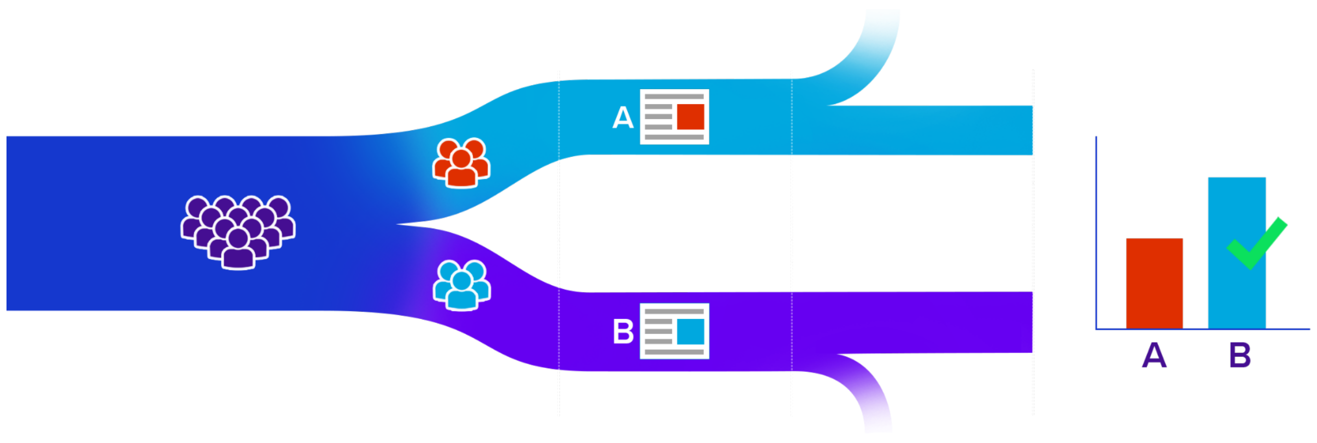
Test duration

The length of time that the A/B test is run. This can vary depending on the sample size and the desired confidence level.

Variance

The degree to which the results of an A/B test vary over time or across different segments of the user base.

Anatomy of an A/B test



Hypothesis
Come up with an idea you want to test

Assignment
Randomly split your audience into persistent groups

Variations
Create and show different experiences to each group

Tracking
Record events and behaviors of the two groups

Results
Use statistics to determine if the differences in behavior are significant

Hypothesis

Good A/B tests, and really any project, starts with a hypothesis about what you're trying to do. A good hypothesis should be as simple, specific, and falsifiable as possible.

A good A/B test hypothesis should be:

- Specific: The hypothesis should clearly state what you want to test and what outcome you expect to see.
- Measurable: The hypothesis should include a metric or metrics that can be used to evaluate the outcome of the test.
- Relevant: The hypothesis should be relevant to your business goals and objectives.
- Clear: The hypothesis should be easy to understand and communicate to others.
- Simple: The fewer variables that are involved in the experiment, the more causality can be implied in the results.
- Falsifiable: The hypothesis should be something that can be tested using an A/B test to determine the validity of the hypothesis.

Overall, a good A/B test hypothesis should be a clear statement that identifies a specific change you want to make and the expected impact on a measurable outcome, while being grounded in data and relevant to your business goals.

Audience and Assignments

Choose the audience for your experiment. To increase the detectable effect of your experiment, the audience you choose should be as close to the experiment as possible. For example, if you're focusing on a new user registration form, you should select as your audience just unregistered users. If you were to include all users, you would have users who could not see the experiment, which would increase the noise and reduce the ability to detect an effect. Once you have selected your audience, you will randomize users to one variation or another.

Variations

An A/B test can include as many variations as you like. Typically the A variation is the control variation. The variations can have as many changes as you like, but the more you change the less certain you can be what caused the change.

Results

With A/B testing we use statistics to determine if the effect we measure on a metric of interest is significantly different across variations. The results of an A/B test on a particular metric can have three possible outcomes: win, loss, or inconclusive. With GrowthBook we offer two different statistical approaches, Frequentist and Bayesian. By default, GrowthBook uses Bayesian statistics. Each method has their pros and cons, but both will provide you with evidence as to how each variation affected your metrics.

Experimentation Basics

Typical success rates

A/B testing can be incredibly humbling—one quickly learns how often our intuition about what will be successful with our users is incorrect. Industry wide average success rates are only about 33%. ⅓ of the time our experiments are successful in improving the metrics we intended to improve, ⅓ of the time we have no effect, and ⅓ of the time we hurt those metrics. Furthermore, the more optimized your product is, the lower your success rates tend to be.

But A/B testing is not only humbling, it can dramatically improve decision making. Rather than thinking we only win 33% of the time, the above statistics really show that A/B tests help us make a clearly right decision about 66% of the time. Of course, shipping a product that won (33% of the time) is a win, but so is not shipping a product that lost (another 33% of the time). Failing fast through experimentation is success in terms of loss avoidance, as you are not shipping products that are hurting our metrics of interest.

Experiment power

With A/B testing, power analysis refers to whether a test can reliably detect an effect. Specifically, it is often written as the percent of the time a test would detect an effect of a given size with a given number of users. You can also think of the power of a test with respect to the sample size. For example: "How many times do I need to toss a coin to conclude it is rigged by a certain amount?"

Minimal Detectable Effect (MDE)

Minimal Detectable Effect is the minimum difference in performance between the control and treatment groups that can be detected by the A/B test, given a certain statistical significance threshold and power. The MDE is an important consideration when designing an A/B test because if the expected effect size is smaller than the MDE, then the test may not be able to detect a significant difference between the groups, even if one exists. Therefore, it is useful to calculate the MDE based on the desired level of statistical significance, power, and sample size, and ensure that the expected effect size is larger than the MDE in order to ensure that the A/B test is able to accurately detect the difference between the control and treatment groups.

False Positives (Type I Errors) and False Negatives (Type II Errors)

When making decisions about an experiment, we can say that we made the right decision when choosing to ship a winning variation or shut down a losing variation. However, because there is always uncertainty in the world and we rely on statistics, sometimes we make mistakes.

Generally, there are two kinds of errors we can make: Type I and Type II errors.

Type I Errors: also known as False Positives, these are errors we make when we think the experiment provides us with a clear winner or a clear loser, but in reality the data are not clear enough to make this decision. For example, your metrics all appear to be winners, but in reality the experiment has no effect.

Type II Errors: also known as False Negatives, these are errors we make when the data appear inconclusive, but in reality there is a winner or a loser. For example, you run an experiment for as long as you planned to, and the data aren't showing a clear winner or loser when actually a variation is much better or worse. Type II errors often require you to collect more data or choose blindly rather than provide you with the correct, clear answer

		Actual Results		
		Inconclusive	**Lost**	**Won**
Decision Made	**Inconclusive**	Correct inference	Type II error (false negative)	Type II error (false negative)
	Shut down	Type I error (false positive)	Correct inference	Type I error (false positive)
	Ship	Type I error (false positive)	Type I error (false positive)	Correct inference

P-Value

In frequentist statistics, a p-value is a measure of the evidence against a null hypothesis. The null hypothesis is the hypothesis that there is no significant difference between two groups, or no relationship between two variables. In the context of A/B testing, the p-value is a statistical measure that indicates whether there is a significant difference between two groups, A and B.

The p-value is the probability of observing a difference as extreme or more extreme as your actual difference, given there is actually no difference between groups. If the p-value is less than a predetermined level of significance (often 0.05), the result is deemed to be statistically significant as the difference is not likely due to chance.

For example, let's say that you conduct an A/B test in which you randomly assign users to either group A (the control group) or group B (the experimental group). You measure a specific metric such as conversion rate for each group, and you calculate the p-value to test the hypothesis that there is no difference between the two groups. If the p-value is less than 0.05, the observed difference conversion rate between the two groups is unlikely if there wasn't truly a difference in groups; we say the effect is statistically significant and likely not due to chance.

It's important to note that p-value alone cannot determine the importance or practical significance of the findings. Additionally, it's essential to consider other factors such as effect size, sample size, and study design when interpreting the results.

A/A Tests

A/A testing is a form of A/B testing in which instead of serving two different variations, two identical versions of a product or design are tested against each other. In A/A testing, the purpose is not to compare the performance of the two versions, but rather to check the consistency of the testing platform and methodology.

The idea behind A/A testing is that if the two identical versions of the product or design produce significantly different results, then there may be an issue with the testing platform or methodology that is causing the inconsistency. By running an A/A test, you can identify and address any potential issues before running an A/B test, which can help ensure that the results of the A/B test are reliable and meaningful.

A/A testing is a useful tool for ensuring the accuracy and reliability of A/B tests, and can help improve the trust in the platform, and faith in the quality of the insights and decisions that are based on the results of these tests.

Interaction effects

When you run more than one test at a time, there is a chance that the tests may interfere with each other. For example, you could have two tests that change the price on two different parts of your product. Some combinations of the two experiments can cause users to see two different prices confused and lose trust in your product. This is an extreme example.

In reality, meaningfully interacting experiments are very rare, and you are better served by running more tests than worrying about unlikely interactions. You can look for interaction effects in the data after the fact (GrowthBook is working on a feature for this).

Novelty and Primacy Effects

Novelty and primacy effects are psychological phenomena that can influence the results of A/B testing.

The novelty effect refers to the tendency of people to react positively to something new and different. In the context of A/B testing, a new design or feature may initially perform better than an existing design simply because it is new and novel. However, over time, the novelty effect may wear off and the performance of the new design may decrease.

The primacy effect refers to the tendency of people to remember and give more weight to information that they encounter first. With A/B testing, this can manifest as an initial reduction in the improvement for metrics as users prefer the original treatment of the product.

One way to mitigate the effects of novelty is to run tests over a longer period of time to allow for the novelty effect to wear off. Another approach is to stagger the rollout of a new design or feature to gradually introduce it to users and avoid a sudden and overwhelming change.

To account for the primacy effect, you can target or segment an experiment to just new users to ensure that they won't be influenced by how things used to work. This can help ensure that the results of the test are truly reflective of user behavior and preferences, rather than the order in which designs were presented.

Experiment Best Practices

Running your first experiment

When you've finished integrating your experimentation platform (which for GrowthBook, is adding the SDK to your code), it's time to start running an experiment. We suggest that you first an A/A test to validate your experimentation implementation is correctly splitting traffic, and producing statistically valid results.

Sample Size

Understanding experiment power and MDE are important to predict how many samples are required. There are numerous online calculators that can be used to help you predict the sample size. Typical rule of thumb for the lowest number of samples required is that you want at least 100 conversion events per variation. So for example if you have a registration page which has a 10% conversion rate, and you have a 2 way (A and B) experiment that is looking to improve the member registrations, you will want to expose the experiment to at least 2,000 people (1000 per variation).

Test Duration

Due to the natural variability in traffic day to day and hour to hour, experimentation teams will often set a minimum test duration within which a test cannot be called. This helps you avoid optimizing a product for just the users that happen to visit when the test is started. For example, if the weekend traffic of your product is different from the traffic during the week, if you started a test on Friday and ended it on Monday, you may not get a complete picture of the impact your changes have to your weekday traffic. Typical test durations are 1 to 2 weeks, and usually care needs to be taken over holidays.

You may also find that a test would need to run for a month or more to get the power required for the experiment. Very long running tests can be hard to justify as you have to keep the variations of the experiment unchanged for duration, and this may limit your team's velocity towards potentially higher impact changes.

Interaction Effects and Mutual Exclusion

When you start having the ability to run a lot of A/B tests, it can be tempting to not want to run tests in parallel in case they have interaction effects. For example you may want to test a change in the CTA button on your purchase page, and also test changing the price. It can be difficult to figure out if any two tests will meaningfully interact, and many will run the tests in serial in an abundance of caution. However, meaningful interactions are actually quite rare, and keeping a higher rate of experimentation is usually more beneficial. You can run analysis after the experiments to see if there were any interaction effects which would change your conclusions. If you need to run mutually exclusive tests, you can use GrowthBook's namespace feature.

Experimentation Frequency

Having a high frequency of A/B testing is important for running a success experimentation program. The main reasons why experimentation frequency is important are:

Maximizing chances: Since success rates are typically low for any given experiment, and large changes are even more rare, by having a high frequency of A/B testing you are maximizing your chance of having impactful experiments.

Continuous improvement: A high frequency of A/B testing allows you to continuously improve your website or application. By testing small changes frequently, you can quickly identify and implement changes that improve user experience, engagement, and conversion rates.

Adaptability: A high frequency of A/B testing allows you to quickly adapt to changes in user behavior, market trends, or other external factors that may impact your website or application. By testing frequently, you can identify and respond to these changes more quickly, ensuring that your site or app remains relevant and effective.

Avoiding stagnation: A high frequency of A/B testing can help you avoid stagnation and complacency. By continually testing and experimenting, you can avoid falling into a rut or becoming overly attached to a specific design or strategy, and instead remain open to new ideas and approaches.

Where Experimentation goes wrong

Multiple Testing Problem

The multiple testing problem refers to the issue that arises when statistical hypothesis testing is performed on multiple variables simultaneously, leading to an increased likelihood of incorrectly rejecting a true null hypothesis (Type I error).

For example, if you test the same hypothesis at a 5% level of significance for 20 different metrics, the probability of finding at least one statistically significant result by chance alone is around 64%. This probability increases as the number of tests performed increases. This math assumes that the metrics are independent from one another, which in most cases for a digital application there will be some interaction between metrics (ie, page views is most likely related to sales funnel starts, or member registration to purchase events)

To address this problem, various multiple comparison correction methods can be used, such as the Bonferroni correction, False Discovery Rate (FDR) correction, or the Benjamini-Hochberg procedure. These methods adjust the significance level or the p-value threshold to account for the increased risk of false positives when multiple comparisons are made.

It's essential to be aware of this issue and select an appropriate correction method when conducting multiple statistical tests to avoid false discoveries and improve the accuracy and reliability of research findings. If you are using a high number of metrics, draw conclusions from the test thoughtfully and if you may consider running a follow up test just to test that one result or metric.

Peeking

The peeking problem refers to the issue of experimenters making decisions about the results of an experiment based on early data. The more often the experiment is looked at, or 'peeked', the higher the false positive rates will be, meaning that the results are more likely to be significant by chance alone. Peeking typically applies to Frequentist statistics, which are statistically valid at their predetermined sample size. However, Bayesian statistics can also suffer from peeking depending on how decisions are made on the basis of Bayesian results.

The peeking problem in A/B testing occurs when the experimenter looks at the data during the experiment and decides to stop the test early based on the observed results, rather than waiting until the predetermined sample size or duration has been reached. This can lead to inflated false positive rates, as the results are more likely to be significant by chance alone if the experimenter stops the test early based on what they see in the data. The more often the experiment is looked at, or 'peeked', the higher the false positive rates will be.

To avoid the peeking problem in A/B testing, it's important to use a predetermined sample size or duration for the experiment and stick to the plan without making any changes based on the observed results. This helps to ensure that the statistical test is valid and that the results are not influenced by experimenter bias.

Another way to avoid the peeking problem in A/B testing is to use a statistical engine that is less susceptible to peeking, like a Bayesian with custom priors, or to use a method that accounts for peeking like Sequential testing.

Problems with client side test

Client-side A/B testing is a technique where variations of a web page or application are served to users via JavaScript on the user's device, without requiring any server-side code changes. This technique can offer a fast and flexible way to test different variations of a website or application, but it can also present some potential problems, one of which is known as "flickering."

Flickering is a problem that can occur when the A/B test is implemented in a way that causes the user interface to render the original version, then flash or flicker as the variations are loaded. This can happen when the A/B test code is slow to load or when the A/B testing library is lacking performance. As a result, the user may see the original version of the page briefly before it is replaced with one of the variations being tested, leading to a jarring and confusing user experience. This flickering can lead to inaccurate or unreliable test results. Rather counterintuitively, flickering may have a positive effect on the results, sometimes the flashing may draw a users attention to that variation, and cause an inflation in the effect.

To avoid flickering in client-side A/B testing, it is important to implement the test code in a way that minimizes the delay between the original page and the variations being tested. This may involve preloading the test code or optimizing the code for faster loading times. GrowthBook's SDKs are built for very high performance, and allow you to use client side A/B testing code inline, so there are no blocking 3rd party calls.

Redirect tests

Redirect-based A/B testing is a technique where users are redirected to different URLs or pages based on the A/B test variation they are assigned to. While this technique can be effective in certain scenarios, it can also present several potential problems that should be considered before implementation.

SEO: Redirects can negatively impact SEO, as search engines may not be able to crawl the redirected pages or may see them as duplicate content. This can result in lower search engine rankings and decreased traffic to the site.

Load times/User experience: Redirects can increase page load times, as the browser has to make an additional HTTP request to load the redirect page. This can result in slower load times, which can impact user experience, conversion rates, and A/B test outcomes.

Data accuracy: Redirects can also impact the accuracy of the test results, as users may drop off or exit the site before completing the desired action due to a slower load time or confusing user experience. It can also be harder technically to fire the tracking event, causing a loss in data.

To mitigate these problems, it's important to carefully consider whether redirect-based A/B testing is the most appropriate technique for your specific use case. If you do choose to use redirects, it's important to implement them correctly and thoroughly test them to ensure that they do not negatively impact user experience or test results. Additionally, it may be helpful to use other techniques such as server-side testing or client-side testing to supplement redirect-based testing and ensure the accuracy and reliability of the test results like testing on the edge or using middleware to serve different pages.

Semmelweis Effect

The Semmelweis effect refers to the tendency of people to reject new evidence or information that challenges their established beliefs or practices. It is named after Ignaz Semmelweis, a Hungarian physician who, in the 19th century, discovered that hand washing could prevent the spread of infectious diseases in hospitals. Despite his findings, he was ridiculed and ignored by his colleagues, and it took many years for his ideas to be accepted and implemented.

In the context of A/B testing, the Semmelweis effect can manifest in several ways. For example, a company may have a long-standing belief that a certain design or feature is effective and produces good results, and may not want to experiment with it because everyone knows it ‘correct’. Even if an experiment is run against this entrenched belief, and the results of an A/B test challenge established norms, there may be resistance to accept the new evidence and change the established practice.

To avoid the Semmelweis effect in A/B testing, it is important to approach experimentation with an open mind and a willingness to challenge established beliefs and practices. It is crucial to let the data guide decision-making and to be open to trying new things, even if they go against conventional wisdom or past practices. It is also important to regularly review and evaluate the results of A/B tests to ensure that the company's beliefs and practices are aligned with the latest evidence and insights, and haven’t changed over time.

Confirmation Bias

Confirmation bias refers to the tendency to favor information that confirms our preexisting beliefs and to ignore or discount information that contradicts our beliefs. In the context of A/B testing, confirmation bias can lead to flawed decision-making and missed opportunities for optimization.

For example, if a company believes that a certain website design or feature is effective, they may only run A/B tests that confirm their beliefs and ignore tests that challenge their beliefs. This can lead to a bias towards interpreting data in a way that supports preexisting beliefs, rather than objectively evaluating the results of the tests. Or a PM may believe a new version of their product will be superior, and only acknowledge evidence that confirms this belief.

Confirmation bias can also manifest in the way tests are designed and implemented. If a company designs an A/B test in a way that biases the results towards a particular outcome, such as by using a biased sample or by selecting a suboptimal metric to measure success, it can lead to misleading results that confirm preexisting beliefs.

To avoid confirmation bias in A/B testing, it is important to approach experimentation with an open and objective mindset. This involves being willing to challenge preexisting beliefs (Semmelweis) and being open to the possibility that the data may contradict those beliefs. It also involves designing tests in a way that is unbiased and that measures the most relevant and meaningful metrics to evaluate success. Having multiple stakeholders review and evaluate the results of A/B tests can help ensure that decisions are based on objective data, rather than personal biases or beliefs.

Trustworthiness

When experiment results challenge existing norms or an individual's beliefs, it can be easy to blame the data. For this reason, having a trustworthy A/B testing platform is extremely important. There must be ways to audit the results, and look into if there was any systemic or specific problem affecting the results of the experiment. Running A/A tests can help build trust that the platform is working correctly. Trust in an experimentation platform is built over time, and care must be taken to not just dismiss results that are counterintuitive.

Twyman's Law

Twyman's law is a principle in statistics that states that any data that is measured and collected will contain some degree of error, and that this error is an inherent part of the data. It is named after the British statistician Maurice G. Kendall Twyman.

In the context of A/B testing, Twyman's law suggests that there will always be some level of variability or uncertainty in the results of an A/B test due to factors such as random chance, sample size, or measurement error. It is often phrased as:

> Any data or figure that looks interesting or different is usually wrong

If you notice a particularly large or unusual change in the results of an experiment, it is more likely to be the result of a problem with the data or an implementation than an actual result. Before you share the results, make sure that the effects are not the result of an error.

Goodhart’s Law

Goodhart's law is a concept in economics that states that when a measure becomes a target, it ceases to be a good measure. In other words, once a metric becomes the sole focus of attention and effort, it loses its value as an indicator of the desired outcome.

When it comes to A/B testing, Goodhart's law can apply in several ways. For example, if a specific metric such as click-through rate or conversion rate becomes the sole focus of an A/B test, it can lead to unintended consequences such as artificially inflating the metric while neglecting other important aspects of the user experience. This can happen because individuals or teams may optimize for the metric being measured rather than focusing on the broader goals of the A/B test, such as improving user engagement or increasing revenue.

To avoid the negative effects of Goodhart's law in A/B testing, it is important to choose the right metrics to track and analyze, and to use a variety of metrics to evaluate the effectiveness of the test. It is also important to keep in mind the broader goals of the test and to avoid tunnel vision on any one metric. Goodhart’s law is more likely to happen when you are using proxy metrics, instead of the real KPIs you’re trying to improve - an example of this might be items added to a cart being used as a proxy for purchases. Also If the proxy metric is not strongly causally linked to the target metric, pressing hard on the proxy may have no effect on the goal metric, or might actually cause the correlation to break.

Simpon’s Paradox

Simpson's paradox is a statistical phenomenon where a trend or pattern appears in different groups of data but disappears or reverses when the groups are combined. In other words, the overall result may be opposite to what the individual subgroups suggest.

This paradox can arise when a confounding variable (a variable that affects both the independent and dependent variables) is not taken into account while analyzing the data.

Simpson's paradox was famously observed at the University of California, Berkeley in 1973, where it had implications for gender discrimination in graduate school admissions.

At the time, it was observed that although the overall admission rate for graduate school was higher for men than for women (44% vs. 35%), when the admission rates were broken down by department, the reverse was true for many of the departments, with women having a higher admission rate than men in each department. In the Department of Education, for example, women had a 77% admission rate compared to men's 62% admission rate.

The paradox was resolved by examining the application data more closely and considering the impact of an important confounding variable, which was the choice of department. It was discovered that women were more likely to apply to departments that were more competitive and had lower admission rates, while men were more likely to apply to less competitive departments with higher admission rates.

When the data was reanalyzed, taking into account the departmental differences in admission rates, it was found that women actually had a slightly higher overall admission rate than men, suggesting that there was no discrimination against women in the admissions process.

This case study illustrates how Simpson's paradox can occur due to the influence of confounding variables, and how it can lead to misleading conclusions if not properly accounted for in the analysis. To avoid the Simpson's paradox in experimentation, it is essential to analyze the data by considering all relevant variables and subgroups. It is crucial to ensure that the experimental groups are similar in terms of demographics and behavior, and to use statistical techniques that account for confounding variables.

Ethical considerations

Experimentation judges the outcome of changes by looking at the impact it has on some set of metrics. But the seeming objectivity of the results can hide problems. The simplest way this can go wrong is if your metrics are tracking the wrong things, in which case you'll have garbage in and garbage out. But it is also possible for the metrics to not capture harm that is being done to some subsets of your population.

Experimentation results work on averages, and this can hide a lot of system biases that may exist. There can be a tendency for algorithmic systems to "learn" or otherwise encode real-world biases in their operation, and then further amplify/reinforce those biases.

Product design has the potential to differentially benefit some groups of users more than others; It is possible to measure this effect and ensure that results account for these groups. Sparse or poor data quality that leads to objective-setting errors and system designs that lead to suboptimal outcomes for many groups of end users. One company that does this very well is the team at LinkedIn, you can read about their approach here.

Running an Experiment in GrowthBook

The Experiments section in GrowthBook is all about analyzing raw experiment results in a data source.

Before analyzing results, you need to actually run the experiment. This can be done in several ways:

- Feature Flags (most common)
- Running an inline experiment directly with our SDKs
- Our Visual Editor (beta)
- Your own custom variation assignment / bucketing system

When you go to add an experiment in GrowthBook, it will first look in your data source for any new experiment ids and prompt you to import them. If none are found, you can enter the experiment settings yourself.

Experiment Splits

When you run an experiment, you need to choose who will get the experiment, and what percentage those users should get each variation. In GrowthBook, we allow you to pick overall exposure percentage, as well as customize the split per variation. Yor can also target an experiment at just some attribute values.

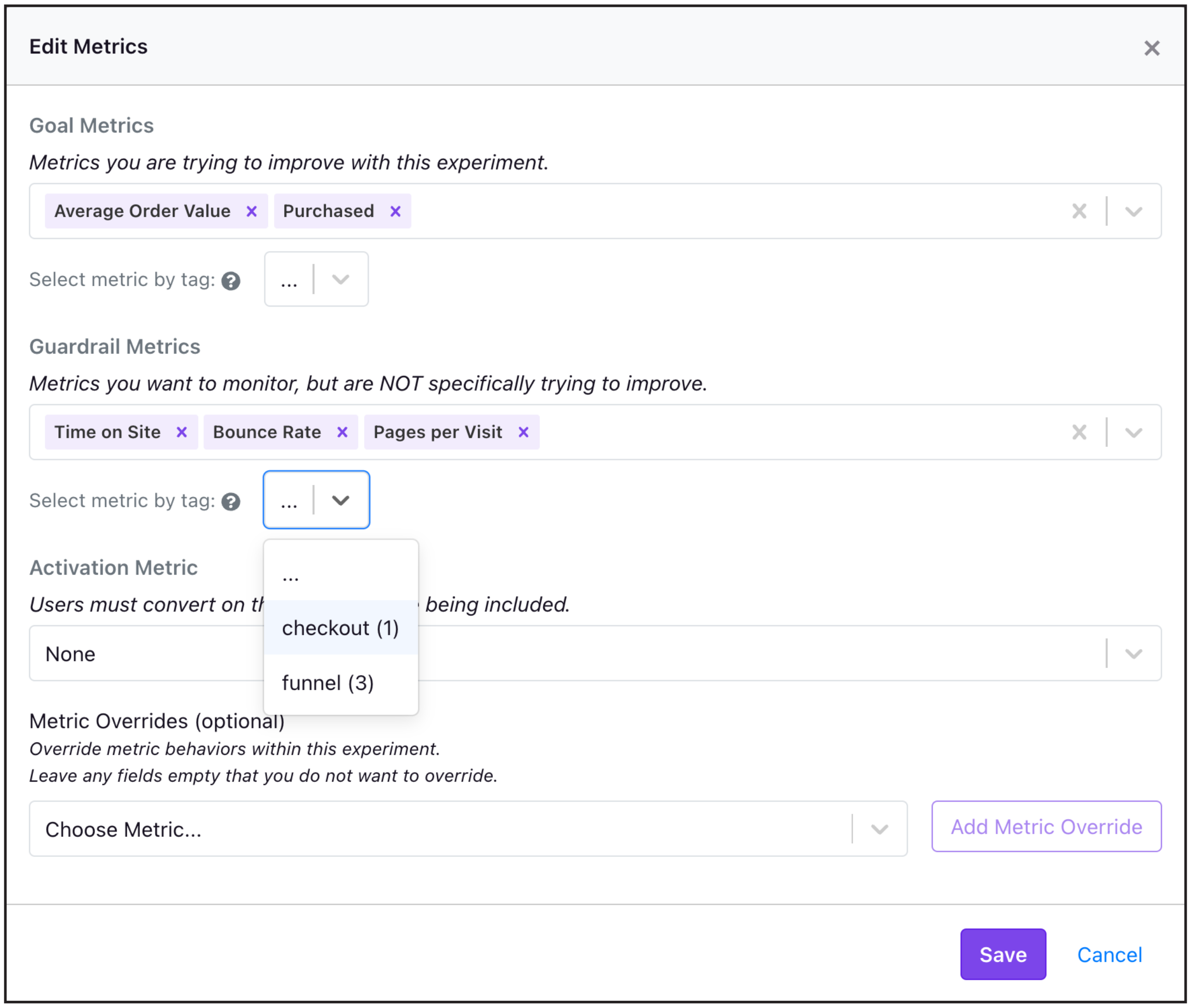
GrowthBook uses deterministic hashing to do the assignment. That means that each user's hashing attribute (usually user id), and the experiment name, are hashed together to get a number from 0 to 1. This number will always be the same for the same set of inputs.

There is quite often a need to de-risk a new A/B test by running the control at a higher percentage of users than the new variation, for example, 80% of users get the control, and 20% get the new variation. To solve this case, we recommend keeping the experiment spits equal, and adjusting the overall exposure (ie, 20% exposure, 50/50 on each variation, so each variation gets 10%). This way the overall exposure can be ramped up (or down) without having any users potentially switch variations.

Metric selection

GrowthBook lets you choose goal metrics and guard rail metrics. Goal metrics are the metrics you're trying to improve or measure the impact of the change of your experiment. Guard rail metrics are metrics you're not necessarily trying to improve, but you don't want to hurt. With goal metrics we show full statistical changes on the metrics. Guard rail metrics we only show the chance of that metric being worse- if there is a significant chance that the guard rail metric is worse, it will be shown in red, otherwise it will be green.

It is best to pick metrics for your experiment that are as close to your treatment as possible. For example, if you're trying to improve product reviews, you can add product reviewMetrics that are close


Edit Metrics

Goal Metrics

Metrics you are trying to improve with this experiment.

Average Order Value × Purchased ×

Select metric by tag: ...

Guardrail Metrics

Metrics you want to monitor, but are NOT specifically trying to improve.

Time on Site × Bounce Rate × Pages per Visit ×

Select metric by tag: ...

Activation Metric

Users must convert on th... being included.

None

Metric Overrides (optional)

Override metric behaviors within this experiment.
Leave any fields empty that you do not want to override.

Choose Metric...

For ease of adding metrics, commonly used sets of metrics can be added by tag.

With GrowthBook, Goal and guardrail metrics can be added retroactively to experiments, as long as the data exists in your data warehouse. This allows you to reprocess old experiments if you add new metrics or redefine a metric.

Activation metrics

Assigning your audience to the experiment should happen as close to the test as possible to reduce noise and increase power. However, there are times when running an experiment requires that users be bucketed into variations before knowing that they are actually being exposed to the variation. One common example of this is with website modals, where the modal code needs to be loaded on the page with the experiment variations, but you're not sure if each user will actually see the modal. With activation metrics you can specify a metric that needs to be present to filter the list of exposed users to just those with that event.

Sample sizes

When running an experiment you select your goal metrics. Getting enough samples depends on the size of the effect you're trying to detect. If you think the experiment will have a large effect, the smaller total number of events you need to collect. GrowthBook allows users to set a minimum sample size for each metric where we will hide results before that threshold is reached to avoid premature peeking.

Test Duration

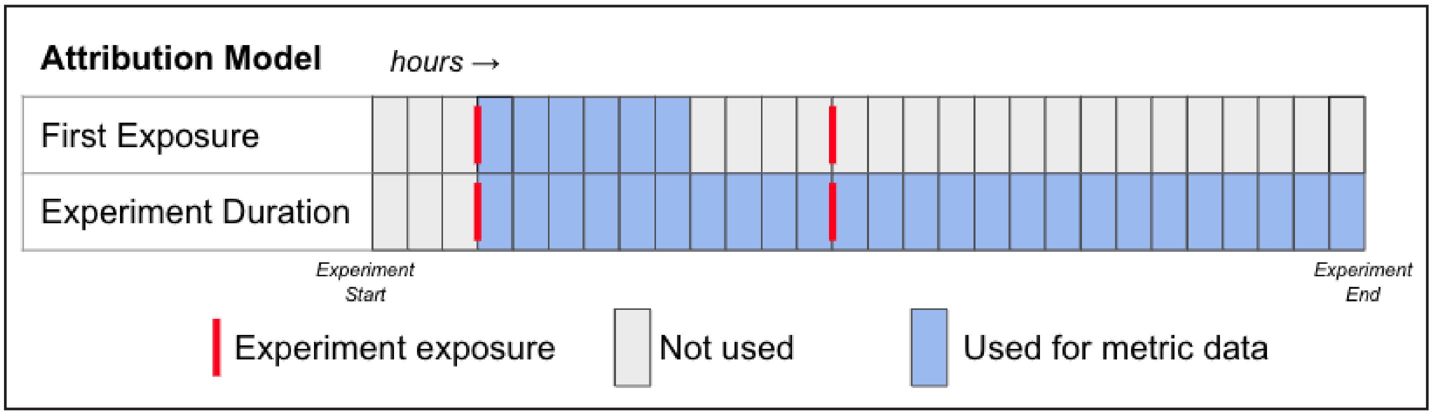
We recommend running an experiment for at least 1 or 2 weeks to capture variations in your traffic. Before a test is significant, GrowthBook will give you an estimated time remaining before it reaches the minimum thresholds.

Attribution models / Conversion Windows

A lot can happen between when a user is exposed to an experiment, and when a metric event is triggered. How you want to attribute that conversion event to the experiment is adjustable within GrowthBook. Once a user enters an experiment for the first time, the attribution model determines which subsequent conversion events are included in the analysis. GrowthBook lets you choose between two attribution models, first exposure, and experiment duration.

First Exposure - Only include events that fall within the configured metric conversion window. For example, if the metric's conversion window is set to 72 hours, any conversion that happens after that is ignored.

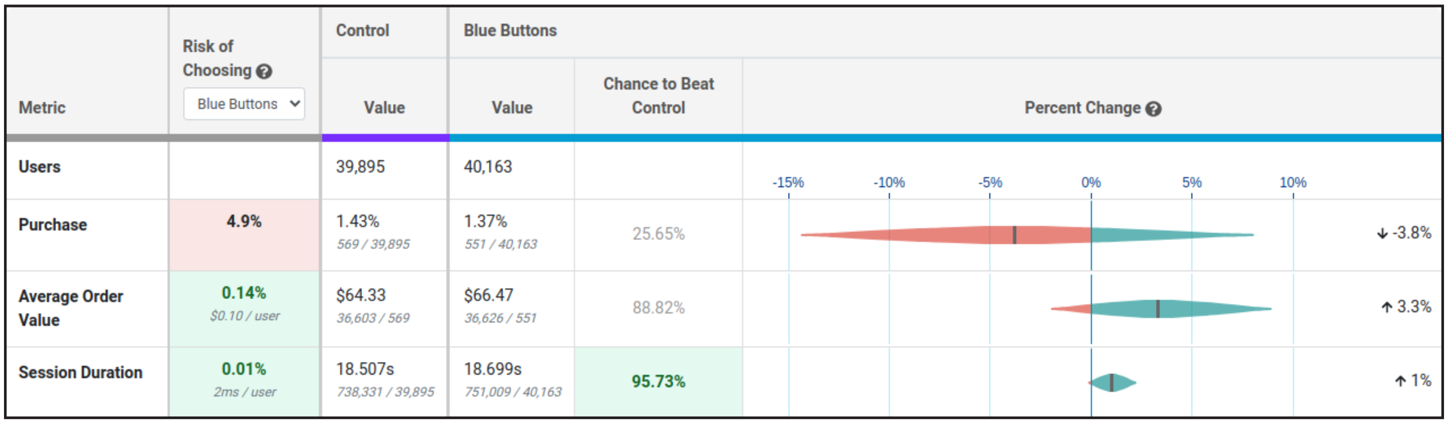
Experiment Duration - Include all events that happen while the experiment is running, no matter what the metric's conversion window is set to.



Understanding results

Bayesian Results

In GrowthBook the experiment results will look like this.



Metric	Risk of Choosing (Blue Buttons)	Control	Blue Buttons		
		Value	Value	Chance to Beat Control	Percent Change
Users		39,895	40,163		-15% -10% -5% 0% 5% 10%
Purchase	4.9%	1.43% *569 / 39,895*	1.37% *551 / 40,163*	25.65%	↓ -3.8%
Average Order Value	0.14% *$0.10 / user*	$64.33 *36,603 / 569*	$66.47 *36,626 / 551*	88.82%	↑ 3.3%
Session Duration	0.01% *2ms / user*	18.507s *738,331 / 39,895*	18.699s *751,009 / 40,163*	95.73%	↑ 1%

Each row of this table is a different metric.

Risk tells you how much you are predicted to lose if you choose the selected variation as the winner and you are wrong. Anything highlighted in green indicates that the risk is very low and it may be safe to call the experiment. You can use the dropdown to see the risk of choosing a different winner if you have multiple variations. Risk thresholds are adjustable per metric.

Value is the conversion rate or average value per user. In small print you can see the raw numbers used to calculate this.

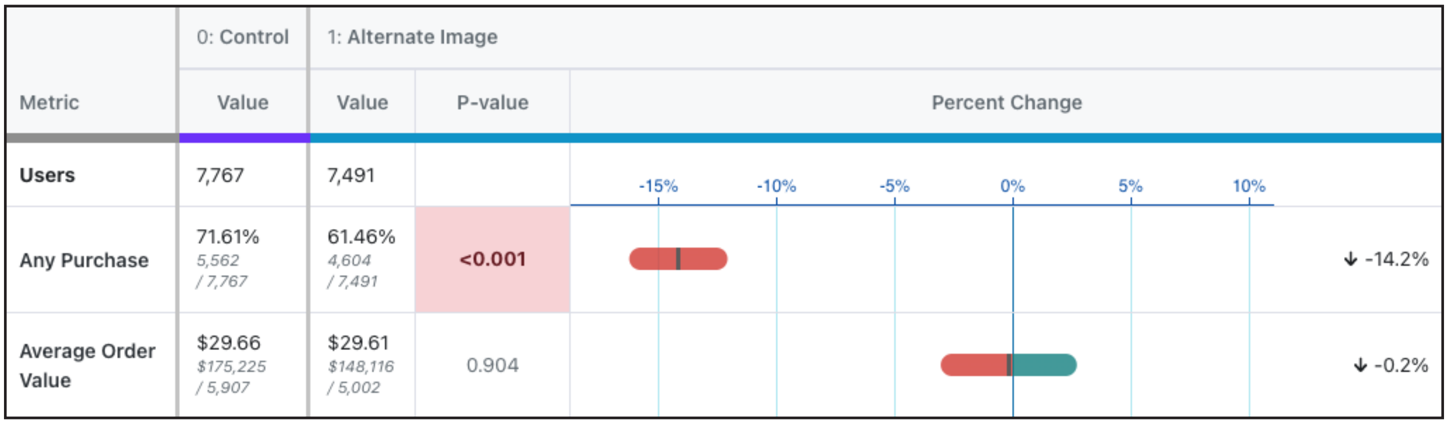
Chance to Beat Control tells you the probability that the variation is better. If you are familiar with Frequentist statistics, you can consider this value 1 - the p value. Anything above the threshold (which by default is set to 95%) is highlighted green indicating a very clear winner. Anything below the threshold (5% by default) is highlighted red, indicating a very clear loser. Anything in between is gray indicating it's inconclusive. If that's the case, there's either no measurable difference or you haven't gathered enough data yet.

Percent Change shows how much better/worse the variation is compared to the control. It is a probability density graph and the thicker the area, the more likely the true percent change will be there. As you collect more data, the tails of the graphs will shorten, indicating more certainty around the estimates.

Frequentist Results

You can also choose to analyze results using a Frequentist engine that conducts simple t-tests for differences in means and displays the commensurate p-values and confidence intervals.

If you selected the "Frequentist" engine, when you navigate to the results tab to view and update the results, you will see the following results table:



	0: Control	1: Alternate Image			
Metric	Value	Value	P-value	Percent Change	
Users	7,767	7,491			
Any Purchase	71.61% 5,562 / 7,767	61.46% 4,604 / 7,491	<0.001		↓ -14.2%
Average Order Value	$29.66 $175,225 / 5,907	$29.61 $148,116 / 5,002	0.904		↓ -0.2%

Everything is the same as above except for three key changes:

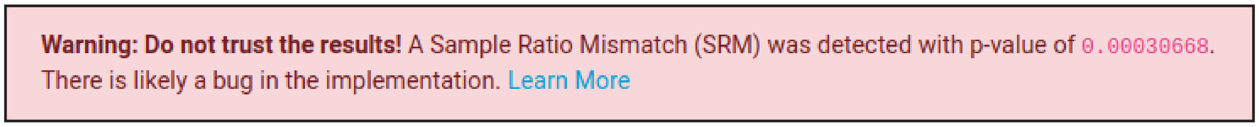
- There is no longer a risk column, as the concept is not easily replicated in frequentist statistics.
- The Chance to Beat Control column has been replaced with the P-value column. The p-value is the probability that the percent change for a variant would have been observed if the true percent change were zero. When the p-value is less than the threshold (default to 0.05) and the percent change is in the preferred direction, we highlight the cell green, indicating it is a clear winner. When the p-value is less than the threshold and the percent change is opposite the preferred direction, we highlight the cell red, indicating the variant is a clear loser on this metric.
- We now present a 95% confidence interval rather than a posterior probability density plot.

Data quality checks

GrowthBook performs automatic data quality checks to ensure the statistical inferences are valid and ready for interpretation.

Sample Ratio Mismatch (SRM)

Every experiment automatically checks for a Sample Ratio Mismatch and will warn you if found. This happens when you expect a certain traffic split (e.g. 50/50) but you see something significantly different (e.g. 46/54). We only show this warning if the p-value is less than 0.001, which means it's extremely unlikely to occur by chance.


Warning: Do not trust the results! A Sample Ratio Mismatch (SRM) was detected with p-value of `0.00030668`. There is likely a bug in the implementation. Learn More

Like the warning says, you shouldn't trust the results since they are likely misleading. Instead, find and fix the source of the bug and restart the experiment.

Multiple Exposures

We also automatically check each experiment to make sure that too many users have not been exposed to multiple variations of a single experiment. This can happen if the hashing attribute is different from the assignment id used in the report, or for implementation problems.

Minimum Data Thresholds

You can set thresholds per metric to make sure people viewing the results aren't drawing conclusions too early (e.g. when it's 5 vs 2 conversions)

Variation Id Mismatch

GrowthBook can detect missing or improperly-tagged rows in your data warehouse

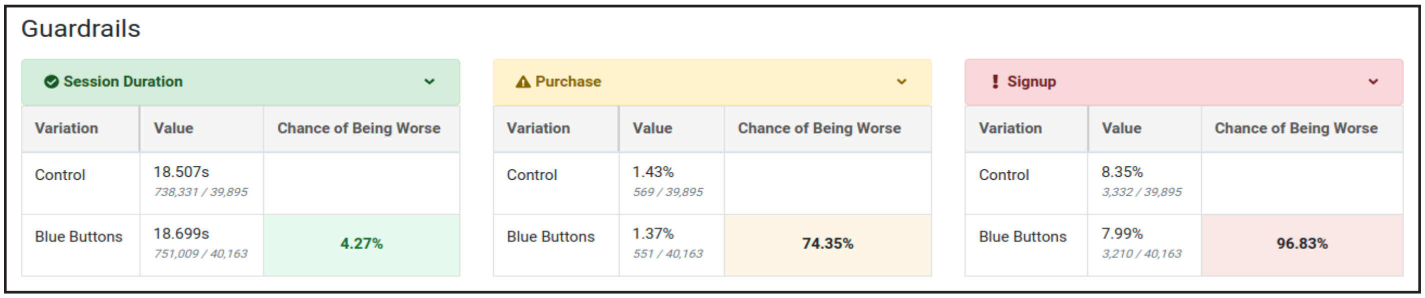
Suspicious Uplift Detection

You can set thresholds per metric for a maximum percent change. When a metric results is above this, GrowthBook will show an alert. Large uplifts may indicate a bug.

Guardrails

metrics are ones that you want to keep an eye on, but aren't trying to specifically improve with your experiment. For example, if you are trying to improve page load times, you may add revenue as a guardrail since you don't want to inadvertently harm it.

Guardrail results show up beneath the main table of metrics and you can click on one to expand it and show more info. They are colored based on "Chance of Being Worse", which is just the complement of "Chance to Beat Control". If there are more than 2 variations, the max value is used to determine the overall color. A "Chance of Being Worse" less than 65% is green and of no concern. Between 65% and 90% is yellow and should be watched as more data comes in. Above 90% is red and you may consider stopping the experiment. If we don't have enough data to accurately predict the "Chance of Being Worse", we will color the metric gray.


Guardrails

✓ Session Duration

Variation	Value	Chance of Being Worse
Control	18.507s 738,331 / 39,895	
Blue Buttons	18.699s 751,009 / 40,163	4.27%

⚠ Purchase

Variation	Value	Chance of Being Worse
Control	1.43% 569 / 39,895	
Blue Buttons	1.37% 551 / 40,163	74.35%

! Signup

Variation	Value	Chance of Being Worse
Control	8.35% 3,332 / 39,895	
Blue Buttons	7.99% 3,210 / 40,163	96.83%

If you select the frequentist engine, we instead use yellow to represent a metric moving in the wrong direction at all (regardless of statistical significance), red to represent a metric moving in the wrong direction with a two-sided t-test p-value below 0.05, and green to represent a metric moving in the right direction with a p-value below 0.05. Otherwise the cell is unshaded if the metric is moving in the right direction but not statistically significant at the 0.05 level.

Digging deeper

GrowthBook lets you dig into the results to get a better understanding of the likely effect of your change.

Segmentation

Segments are applied to experiment results to only show users that match a particular attribute. For example, you might have “country” as a dimension, and create a segment for just “US visitors”. In the experiment you can configure the experiment to just look at one particular segment of users.

Dimensions

GrowthBook lets you break down results by any dimension you know about your users. We automatically let you break down by date, and any additional dimensions can be added either with the exposure query, or with custom SQL from the dimension menu. Some examples of common dimensions are “Browser” or “location”.

It can be very helpful to look into how specific dimensions of your users are affected by the experiment. For example, you may discover that a specific browser is underperforming compared to the rest, and this may indicate a bug, or something to investigate further.

The more metrics and dimensions you look at, the more likely you are to see a false positive. If you find something that looks surprising, it's often worth a dedicated follow-up experiment to verify that it's real.

Ad-hoc reports

Experiment reports have a lot of configuration, and sometimes it can be useful to want to adjust these configurations without changing the original report. GrowthBook supports Ad-hoc reports, which are essentially copies of the original report, where you can adjust any of the configuration parameters, such as segments, dates, metrics, even custom SQL to remove outliers. All ad hoc reports created can be shared publicly and live at the bottom of the report, to make sure you capture any derived results.

Deciding A/B test results

When to stop an experiment

When using the Bayesian statistic engine, there are a few methods you can use when stopping a test.

- significance reached on your primary metrics

- metric risk drop below your risk thresholds
- guard rail metrics are not affected

It all depends on what you're trying to do with the experiment. For example, if you'd like to know what impact your change has, you should use the first method. If you're doing a design change, and want to make sure you haven't broken anything on your product, you can use the risk or guard rail approach. You should also make sure that the experiment has run for your minimum test duration (typically 1 or 2 weeks), so that you're not looking at highly skewed sampling.

For Frequentist statistics, you should determine the running time of the experiment and stop the test at that fixed horizon to ensure accurate results (see Peeking).

Interpreting results

It is quite common to have experiment results with mixed results. Deciding on the results of an experiment in these cases may require some interpretation. As a general rule, you should have one goal metric that is the primary metric you're trying to improve, and if this metric is up significantly it is generally straightforward to declare a result. If you have a mix and up and down metrics, the decisions are less clear.

Once you have reached a decision with your experiment, you can click the "mark as finished" link towards the top of the results. This will open a modal where you can document the results, including the result, and observations.

This creates a card on the top of the experiment results with your conclusion.

Please note that currently this marking a test as finished does not stop the test from running. If you are using feature flags to run the experiment, you should also go to the feature and turn off the experiment.

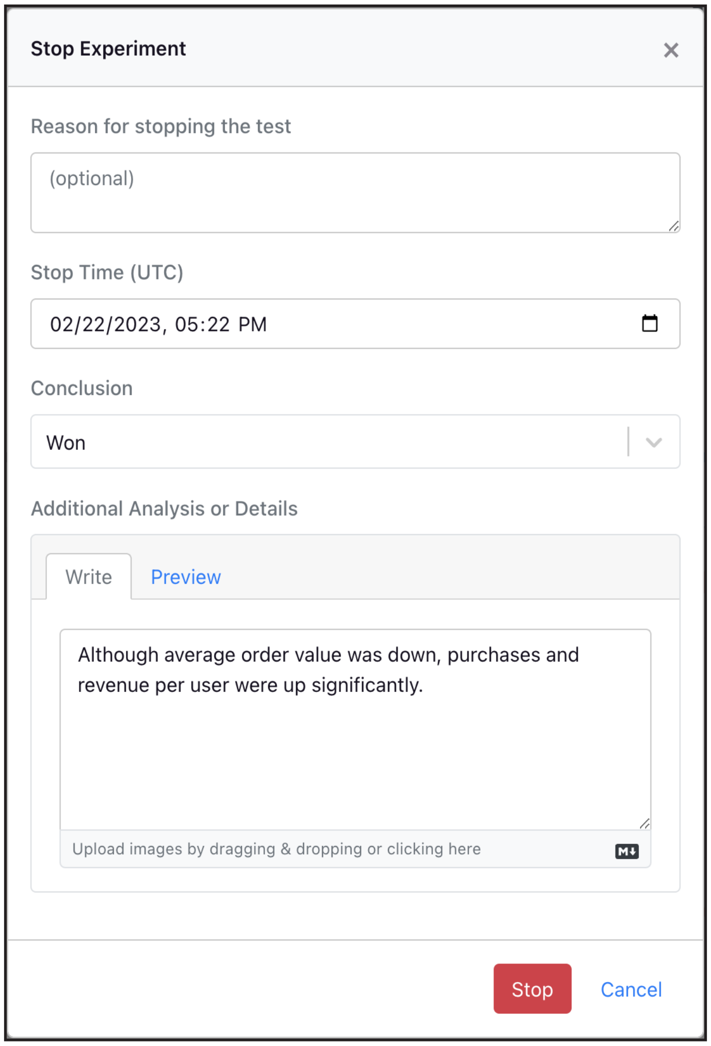
GrowthBook Best Practices

Organization

GrowthBook provides some tools to help you organize your features and experiments and help your program scale.

Environments

In GrowthBook, you can create Environments. Environments are meant to separate how your code is deployed. For example, you might have environments for "Staging", "QA", and "Production".


Stop Experiment

Reason for stopping the test

(optional)

Stop Time (UTC)

02/22/2023, 05:22 PM

Conclusion

Won

Additional Analysis or Details

Although average order value was down, purchases and revenue per user were up significantly.

Upload images by dragging & dropping or clicking here

Projects

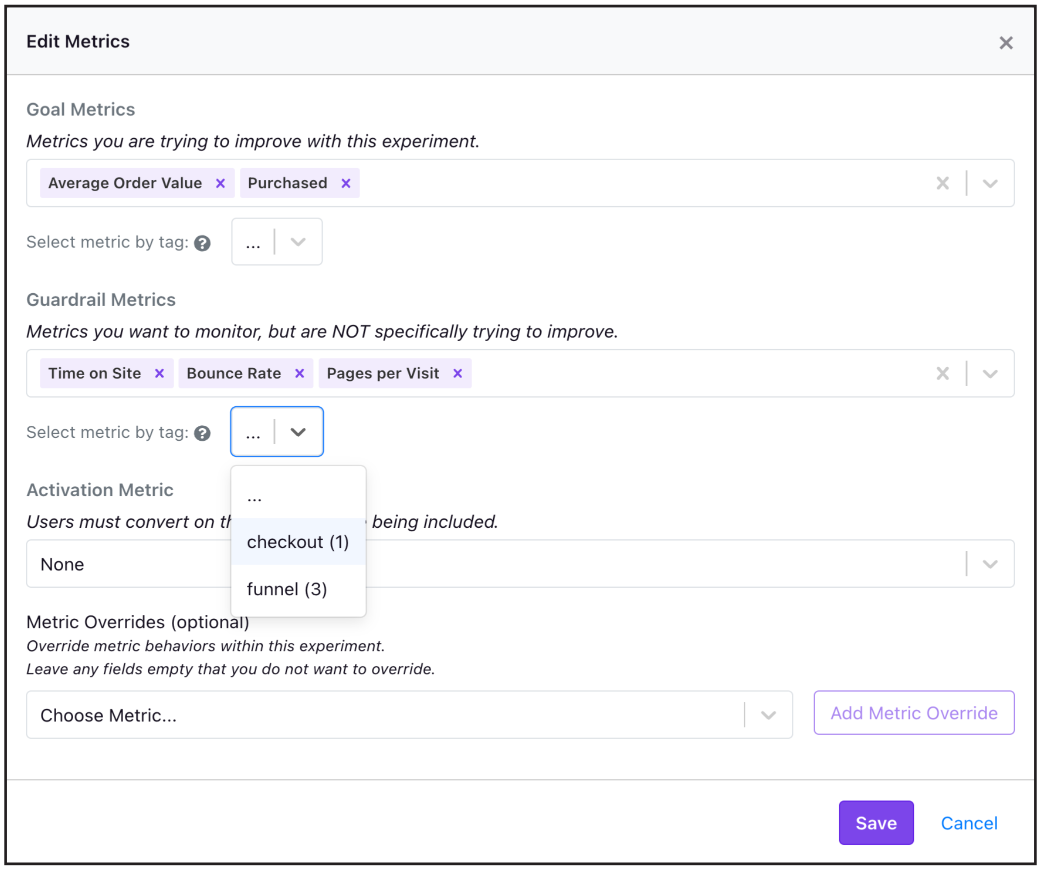
Projects are a great way to organizationally separate features and experiments by team or product feature. For example, you could have a project “front-end” and one for “back-end”, or by team like “Growth” and “API”. In this way you can help make sure the features and experiments are easier to see when there are a lot of teams using GrowthBook.

To help keep feature payloads smaller, the SDK endpoint where the feature definitions are returned can be scoped to each project. If using Projects based on features or area of your product, you can use this feature to only return features which pertain to that area. For example, if you use “mobile” and “website” as projects, you can add the project scope to only return features for the mobile app, as these are likely to use different code than the website, and you don’t want to expose features unnecessarily.

Tags

Another way to organize GrowthBook is with tags. With tags, you can quickly filter lists, and select metrics. For example, if you tagged all metrics to do with your checkout flow with the tag “checkout”, you can quickly see this in the list by clicking on ‘filter by tags’ on the metric list, and also quickly add these metrics to any experiment.

Metrics with tags can be used to quickly add all those metrics to an experiment. When creating an experiment or editing the metrics, there is a section titled “Select metric by tag” which will let you add all the metrics by the tag name.


Edit Metrics

Goal Metrics

Metrics you are trying to improve with this experiment.

Average Order Value, Purchased

Select metric by tag: ...

Guardrail Metrics

Metrics you want to monitor, but are NOT specifically trying to improve.

Time on Site, Bounce Rate, Pages per Visit

Select metric by tag: ...

Activation Metric

Users must convert on th... being included.

None

Metric Overrides (optional)

Override metric behaviors within this experiment.

Leave any fields empty that you do not want to override.

Naming

Because GrowthBook makes it easy to quickly search the list of features and flags, using naming conventions can also be an effective way to organize your project. Consider adopting something like <project scope>_<project name>.

Archiving

For experiment and features, you can archive ones that are unused to help keep the list to just your active work.

GrowthBook provides built in tools to help you share your experiment results with your team.

Sharing

One great way to get fresh ideas and to help experimentation culture is to share your experiment ideas and results. Our preferred way to present your results is with an experiment review meeting. The premise behind these is to talk about the experiment without revealing the results, and to have people guess about the outcome. Specifically, you talk about the hypothesis and observations as to what and why you are testing, and then talk about the metrics you're testing against, and then show screen shots of the variations (if applicable). You can have people vote simply by raising their hand. Once you've had people guess, you reveal the actual results. This is a great way to help build intellectual humility and also collect new ideas.

Experimentation Programs

“Experimentation”, or being more “data driven” can mean a lot of different things for different companies. It can be anything from running 1 test a quarter, to running 10s of thousands of experiments simultaneously. This difference of experimentation sophistication can be thought of with the crawl, walk, run, fly framework.[1]

CRAWL - Basic Analytics

Companies at this stage have added some basic event tracking and are starting to get some visibility into their users behavior. They are not running experiments, but the data is used to come up with insights and potential project ideas.

WALK - Optimizations

After implementing comprehensive event tracking, focus now turns to starting to optimize the user experience on some parts of the product. At this stage, A/B tests may be run manually, limiting the number of possible experiments that can be run. Typically at this stage, depending on the amount of traffic you have, you may be running 1 to 4 tests per month.

RUN - Common Experimentation

As a company realizes that experimentation is really the only way to causally determine the impact of the work they are doing, they will start to ramp up their ability to run A/B tests. This means adopting or building an experimentation platform. With this, all larger changes made to their product are tested, as well they may have a Growth Team that is focused on optimizing parts of their product. At this stage, a company will be running 5 - 100 A/B Tests/Month. This may also include hiring a data team to help with setting up and interpreting the results.

FLY - Ubiquitous Experimentation

For companies that make it to the flying stage, A/B testing becomes the default for every feature. Product teams develop success metrics for all new features, and then they are run as an A/B test to determine if they were successful. At this point, A/B tests are able to be run by anyone in the product and engineering organization. Companies at this stage of ubiquitous experimentation can run anywhere from 100 to 10,000+ A/B Tests/Month.

Measuring Experiment Program Success

Once you have added an experimentation program, teams often look for a way to measure the success of that program. There are a few ways you can use to measure the success of your experimentation program, such as universal holdouts, win rate, experimentation frequency and learning rate. Each of these has their own advantages and disadvantages.

Universal Holdouts

Universal holdouts is a method for keeping a certain percentage of your total traffic from seeing any new features or experiments. Users in a universal holdout will continue to get the control version of every test for an extended period of time, even after an experiment is declared, and then those users

1 From “Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing”, Ron Kohavi, Diane Tang, Ya Xu

are compared to users who are getting all the new features and changes. This effectively gives you a cohort of users that are getting your product as it was, say 6 months ago, and comparing it to all the work you've done since. This is the gold standard for determining the cumulative impact of every change and experiment, however, it has a number of issues.

To make universal holdouts work, you need to keep the code that delivers the old versions running and working on your app. This is often very hard to do. Some changes can have a non zero maintenance cost, block larger migrations, or limit other features until the holdout ends. Also, any bugs that arise that only affect one side of the holdouts (either control or in the variations), can bias the results. Finally, due to the typically smaller size of the universal holdout group, it can take longer for these holdout experiments to reach significant values, unless you have a lot of traffic.

Given the complexity of running universal holdouts, many companies and teams look for other proxy metrics or KPIs to use for measuring experimentation program success.

Win Rate

It can be very tempting to want to measure the experimentation win rate, the number of A/B tests that win over the total number of tests, and optimize your program for the highest win rate possible. However, using this as the KPI for your experiment program will encourage users to not run high risk experiments and creates a perverse incentive for more potentially impactful results (see Goodhart's Law). Win rate can also hide the benefits of not launching a "losing" test, which is also a "win".

Experimentation Frequency

A more useful measure than win rate is optimizing for the number of experiments that are run. This encourages your team to run a lot of tests which increases the chances of any one test producing meaningful results. It may, however, encourage you to run smaller experiments over larger ones, which may not be optimal for producing the best outcomes.

Learning Rate/Learning Quality

Some teams try to optimize for a "learning rate" which is the rate at which you learn something about your product or users through A/B testing. This does not have the frequency or win rate biases, but also is nebulously defined. How do you define learning? Are there different qualities of what you learn?

KPI Effect

If you can pick a few KPIs for your experimentation program, you should be able see the effects of the experiments you run against this. You may not be able to see causality precisely due to the natural variability in the data, and typically small improvements from an A/B test, but by aligning by the graph of this metric to experiments that are run, you may start to see cumulative effects. This is what GrowthBook shows with our North Star metric feature.

Prioritization

Given the typical success rates of experiments, all prioritization frameworks should be taken with a grain of salt. Our preference at GrowthBook is to add as little process as possible and to maximize for a good mix of iterative and innovative ideas.

Iteration vs Innovation

It is useful to think of experiment ideas on a graph with one axis being the effort required and the potential impact on the other. If you divide the ideas into two for high effort/impact and low effort/ impact, you'll end up with the following quadrant.

High effort	Danger	Prioritize
Low effort	Prioritize	Run now
	Low impact	High impact

The low effort, high impact ideas you should be running immediately, and similarly the high effort, low impact ideas you may not want to run at all. But this leaves the other two, low effort but low impact (smaller tests), and high effort high impact ideas (big bets). If you over index for smaller test ideas, you can increase your experimentation frequency, but risk not getting larger gains. If you over index for bigger bets, you decrease your experimentation frequency at the hope of larger returns, at the risk of not achieving the smaller wins which can stack up. You can also consider the smaller tests as being "iterative" and the bigger bets as "innovative".

Finding a good mix of small, iterative tests and bigger bets/innovative tests is the best strategy. What constitutes "good" is up to the team. Some companies will bucket their ideas into these two groups, and then ensure that they are pulling some percentage of ideas from both lists. A healthy mix of large and small ideas are important to a successful experimentation program.

Prioritization frameworks

In the world of A/B testing, figuring out what to test can be particularly challenging. Often prioritization requires a degree of gut instinct which is often incorrect (see success rates). To solve this, some recommend prioritization frameworks, such as ICE and PIE.

Note: Please keep in mind that while these frameworks may be helpful, they can work to give the appearance of objectivity to subjective opinions.

ICE

The ICE prioritization framework is a simple and popular method for prioritizing A/B testing ideas based on their potential impact, confidence, and ease of implementation. Each idea is evaluated on each of these factors and scored on a scale of 1 to 10 and then averaged to determine the overall score for that idea. Here's a brief explanation of the factors:

Impact: This measures the potential impact of the testing idea on the key metrics or goals of the business. The impact score should reflect the expected magnitude of the effect, as well as the relevance of the metric to the business objectives.

Confidence: This measures the level of confidence that the testing idea will have the expected impact.

The confidence score should reflect the quality and quantity of the available evidence, as well as any potential risks or uncertainties.

Ease: This measures the ease or difficulty of implementing the testing idea. The ease score should reflect the expected effort, time, and resources required to implement the idea.

To calculate the ICE score for each testing idea, simply add up the scores for Impact, Confidence, and Ease, and divide by 3:

ICE score = (Impact + Confidence + Ease) / 3

Once all testing ideas have been scored using the ICE framework, they can be ranked in descending order based on their ICE score. The highest-ranked ideas are typically considered the most promising and prioritized for implementation.

PIE

Like the ICE Framework, the PIE framework is a method for prioritizing A/B testing ideas based on their potential impact, importance to the business, and ease of implementation. Each score is ranked on a 10 point scale.

Potential: This measures the potential impact of the testing idea on the key metrics or goals of the business. The potential score should reflect the expected magnitude of the effect, as well as the relevance of the metric to the business objectives.

Importance: This measures the importance of the testing idea to the business. The importance score should reflect the degree to which the testing idea aligns with the business goals and objectives, and how critical the metric is to achieving those goals.

Ease: This measures the ease or difficulty of implementing the testing idea. The ease score should reflect the expected effort, time, and resources required to implement the idea.

To calculate the PIE score for each testing idea, simply multiply the scores for Potential, Importance, and Ease together:

PIE score = Potential x Importance x Ease

Once all testing ideas have been scored using the PIE framework, they can be ranked in descending order based on their PIE score. The highest-ranked ideas are typically considered the most promising and prioritized for implementation.

Experimentation Culture

Adopting experimentation as a key part of being a more data-driven organization has numerous benefits to culture. Specifically around areas of alignment, speed, humility, and collaboration.

Alignment

Adopting a north star metric or KPI that would drive our business success removes a lot of ambiguity about projects because we had clear success metrics. By making sure you have defined success metrics at the start of your planning cycle, you achieve alignment around your goals. This helps reduce the invariable scope creep and pet features from inserting themselves — or at least gives you a framework to say “yes, but not now.” Knowing what success means also allows developers to start integrating the tracking needed to know if the project would be successful from the beginning, which can often be forgotten or only done as an afterthought.

Speed

When adopting an experimentation mindset, the default answer to a difference of opinion becomes "let's test it" instead of long drawn out ego bruising meetings. This helps reduce personal opinions or bias affecting decisions. Quite often decisions in companies without this mindset are made by whomever is the loudest, or the HiPPOs (Highest Paid Person's Opinion). By focusing on which metrics defined success, and defaulting to running an experiment, you can remove the ego from the decision process, and move quickly.

Experimentation can also help increase your product velocity by minimizing the time it takes to determine if your new product or feature has product market fit. Most big ideas can be broken down into a small set of assumptions that, if true, would mean your larger idea may be successful. If you can prove or disprove these ideas, you can move more quickly and not waste time on failing ideas (loss avoidance).

Intellectual humility

AB testing shows us that, in most of the cases, people are bad at predicting user behaviors. When you realize that your opinions may not be correct, you can start to channel your inner Semmelweis and be open to new ideas that challenge any deeply held entrenched norms or beliefs. Having an open mind and intellectual humility for new ideas can make your workplace a more collaborative environment, and produce better products.

Team collaboration

When you are open to new ideas, you can remove the silos that prevent teams from collaborating well. The goal is to produce the best product as measured by a specific set of metrics. With this alignment, and the openness to new ideas, you can dramatically increase collaboration as good ideas come from anywhere.

Conclusion

A/B testing is a critical tool for determining causal impact of the changes you make, as well as optimizing flows. By making informed data-driven decisions, you can improve user experience, increase conversions, and drive growth. With the open source A/B testing tool, GrowthBook, you have a powerful and flexible platform that can help you run experiments quickly and easily.

We hope that this book has given you the knowledge and skills you need to run successful A/B tests and make data-driven decisions. Whether you're a developer, product manager, data scientist, marketer, or business owner, A/B testing can help you achieve your goals and drive growth.

Thank you for reading, and we wish you the best of luck in your A/B testing journey.