Always Valid Inference

Continuous Monitoring of A/B Tests

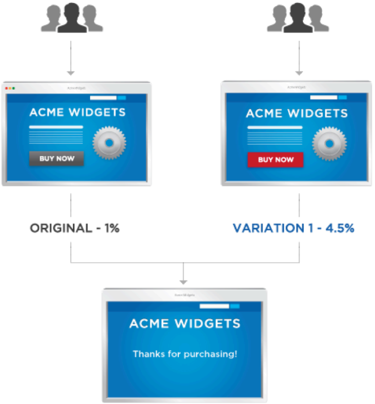
Ramesh Johari | Leo Pekelis | David Walsh
Stanford University / Optimizely
rjohari@stanford.edu

21 September 2016

Background: Online A/B Testing

What is A/B testing?

- *A/B testing* = randomized controlled trials used by technology companies and web applications
- Typical use case: comparing versions of a web page
- Question: ***Does one yield a higher conversion rate?***



How it works

- Visitors are randomized to Variation A (control) or B (treatment).
- Conversion rates tracked in each group.
- Let p_A, p_B be *true* underlying conversion rates in each group.
- Hypothesis test:

  $$H_0 \ : \ \theta = 0 \text{ (Null hypothesis)}$$
  $$H_1 \ : \ \theta \neq 0 \text{ (Alternative hypothesis)}$$

  where $\theta = p_A - p_B$ is the *conversion rate difference*.

How it works

Fixed horizon testing:

- The user **must** set sample size N **in advance**.
- After each new visitor, the interface computes a *p-value*: $p_n = \mathbb{P}($data at least as "extreme" as current sample $|H_0)$.
- Simple decision rule: Reject H_0 if *p-value* $p_N \leq \alpha$.

How it works

Fixed horizon testing:

- The user **must** set sample size N **in advance**.
- After each new visitor, the interface computes a *p-value*: $p_n = \mathbb{P}(\text{data at least as “extreme” as current sample } | H_0)$.
- Simple decision rule: Reject H_0 if *p-value* $p_N \leq \alpha$.

This approach:

- Bounds *Type I error* (false positive probability) at level α.

How it works

Fixed horizon testing:

- The user **must** set sample size N **in advance**.
- After each new visitor, the interface computes a *p-value*: $p_n = \mathbb{P}($data at least as "extreme" as current sample $|H_0)$.
- Simple decision rule: Reject H_0 if *p-value* $p_N \leq \alpha$.

This approach:

- Bounds *Type I error* (false positive probability) at level α.
- Gives optimal power (true positive probability) given α (assuming a UMP hypothesis test is used).

How it works

Fixed horizon testing:

- The user **must** set sample size N **in advance**.
- After each new visitor, the interface computes a *p-value*: $p_n = \mathbb{P}($data at least as "extreme" as current sample $| H_0)$.
- Simple decision rule: Reject H_0 if *p-value* $p_N \leq \alpha$.

This approach:

- Bounds *Type I error* (false positive probability) at level α.
- Gives optimal power (true positive probability) given α (assuming a UMP hypothesis test is used).
- Allows *many* users to draw inferences on the same dashboard, without knowing the details of the experiment.

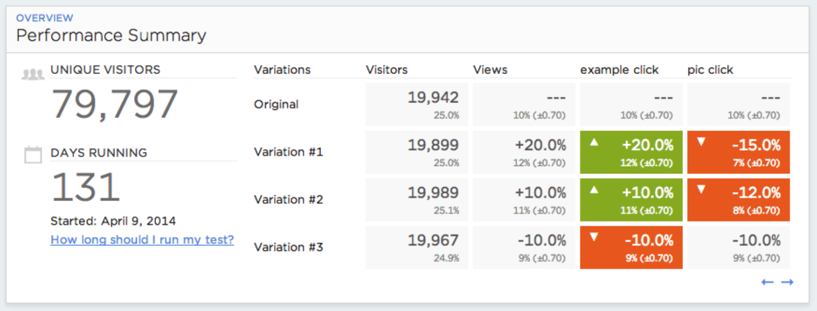
Continuous Monitoring

Continuous monitoring

In practice:

Technology makes it convenient to
continuously monitor tests!

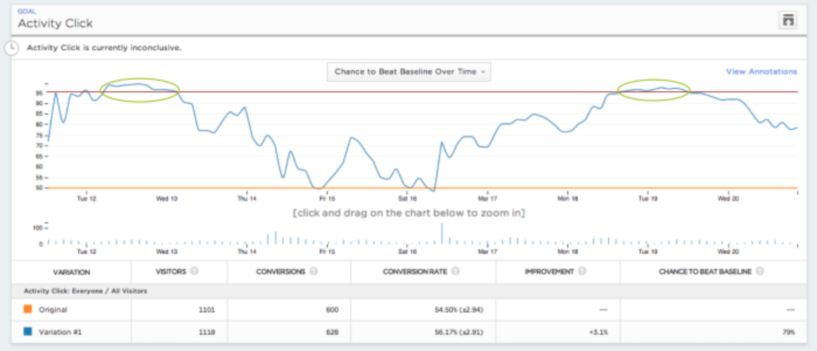
E.g., results matrix:



Variations	Visitors	Views	example click	pic click
Original	19,942 25.0%	--- 10% (±0.70)	--- 10% (±0.70)	--- 10% (±0.70)
Variation #1	19,899 25.0%	+20.0% 12% (±0.70)	+20.0% 12% (±0.70)	-15.0% 7% (±0.70)
Variation #2	19,989 25.1%	+10.0% 11% (±0.70)	+10.0% 11% (±0.70)	-12.0% 8% (±0.70)
Variation #3	19,967 24.9%	-10.0% 9% (±0.70)	-10.0% 9% (±0.70)	-10.0% 9% (±0.70)

The problem with peeking

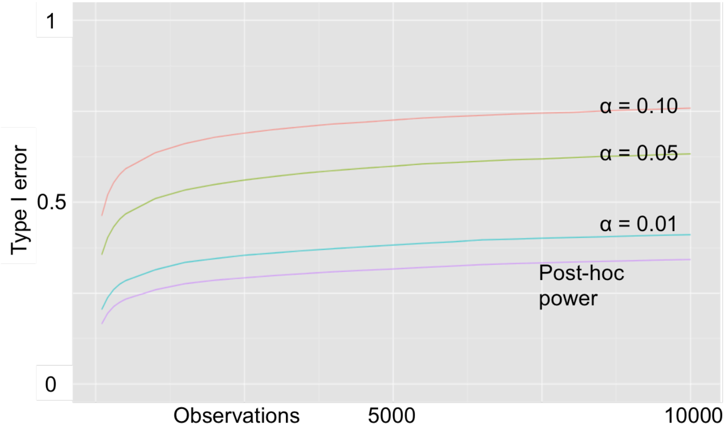
Example: A sample path from an **A/A** test:



The problem with peeking

Unfortunately this *dramatically* inflates Type I errors! In fact, with arbitrarily large horizon, Type I error is **guaranteed**.

Even on finite horizons, Type I errors are highly inflated:



Why?

Why do users continuously monitor?

Because there is value in detecting real effects as quickly as possible, and high opportunity cost in waiting to end a test.

In other words, user are making a dynamic tradeoff between *detection* (higher power) and *run time*.

This is a risk preference that is *not known* to the platform.

Our challenge

Can we:

1. deliver essentially optimal inference (like classical p-values and confidence intervals);
2. in an environment where users continuously monitor experiments;
3. and when the platform does not know the user's priorities regarding run-time and detection in advance?

Our work addresses this challenge.

It was released to Optimizely's entire customer base worldwide in 2015.

The plan

1. *Always valid* p-values: Control Type I error despite continuous monitoring
2. Efficiently trade off power and run-time
3. Implementation in an A/B testing platform
4. Multiple hypothesis testing corrections

Always Valid Statistics

Always valid p-values

Initial goal:

- A user should be able to look at their results **whenever** they want.
- The p-value at that time should give valid type I error control.

Always valid p-values

Definition

A **(fixed-horizon) p-value** process is a (data-dependent) sequence p_n such that for all n and all $x \in [0, 1]$:

$$\mathbb{P}_0(p_n \leq x) \leq x.$$

Always valid p-values

Definition

A **(fixed-horizon) p-value** process is a (data-dependent) sequence p_n such that for all n and all $x \in [0, 1]$:

$$\mathbb{P}_0(p_n \leq x) \leq x.$$

Definition

A p-value process is **always valid** if for any data-dependent stopping time T and all $x \in [0, 1]$:

$$\mathbb{P}_0(p_T \leq x) \leq x.$$

Always valid p-values

Definition

A **(fixed-horizon) p-value** process is a (data-dependent) sequence p_n such that for all n and all $x \in [0, 1]$:

$$\mathbb{P}_0(p_n \leq x) \leq x.$$

Definition

A p-value process is **always valid** if for any data-dependent stopping time T and all $x \in [0, 1]$:

$$\mathbb{P}_0(p_T \leq x) \leq x.$$

Allows user to favorably bias the choice of T based on the data that is seen.

Constructing Always Valid p-values

Sequential tests

Definition

A **sequential test** $\{T_\alpha\}$ is a data-dependent rule for stopping the test and rejecting the null that:

1. stops the test later when α is lower; and
2. stops with probability $\leq \alpha$ when the null is true:

$$\mathbb{P}_0(T_\alpha < \infty) \leq \alpha.$$

Constructing always valid p-values

Theorem

Given a sequential test, define the p-value p_n to be:

the smallest α such that the α-level test would have stopped by observation n.

Then the resulting p-value process is always valid.

Proof of theorem

Step 1. p_n is decreasing.

Proof of theorem

Step 1. p_n is decreasing.

Step 2. Thus p_∞ exists a.s.

Proof of theorem

Step 1. p_n is decreasing.

Step 2. Thus p_∞ exists a.s.

Step 3. For fixed $\alpha > x$, the event $\{ T_\alpha < \infty\}$ contains the event $\{p_\infty \leq x\}$, so:

$$\mathbb{P}_0(p_\infty \leq x) \leq \mathbb{P}_0(T_\alpha < \infty) \leq \alpha.$$

Proof of theorem

Step 1. p_n is decreasing.

Step 2. Thus p_∞ exists a.s.

Step 3. For fixed $\alpha > x$, the event $\{ T_\alpha < \infty \}$ contains the event $\{p_\infty \leq x\}$, so:

$$\mathbb{P}_0(p_\infty \leq x) \leq \mathbb{P}_0(T_\alpha < \infty) \leq \alpha.$$

Step 4. Thus taking $\alpha \to x$, for any stopping time T:

$$\mathbb{P}_0(p_T \leq x) \leq \mathbb{P}_0(p_\infty \leq x) \leq x.$$

Duality

Note that if a user stops the first time that the always valid p-value drops below α, then the stopping time is T_α.

Thus we have a simple decision rule that implements the sequential test.

Power and Run-Time

Power vs. run-time

Recall: users are trying to efficiently trade off power and run-time.

In order to make progress, some user model is needed.

Power vs. run-time

Recall: users are trying to efficiently trade off power and run-time.

In order to make progress, some user model is needed.

We use the following:

1. We choose a method of computing p-values.

Power vs. run-time

Recall: users are trying to efficiently trade off power and run-time.

In order to make progress, some user model is needed.

We use the following:

1. We choose a method of computing p-values.
2. The user observes p-values, and wants to detect whether a nonzero effect exists (reject H_0).

Power vs. run-time

Recall: users are trying to efficiently trade off power and run-time.

In order to make progress, some user model is needed.

We use the following:

1. We choose a method of computing p-values.
2. The user observes p-values, and wants to detect whether a nonzero effect exists (reject H_0).
3. The user stops the first time the p-value falls below α, up to a maximum run time of M.

Power vs. run-time

Recall: users are trying to efficiently trade off power and run-time.

In order to make progress, some user model is needed.

We use the following:

1. We choose a method of computing p-values.
2. The user observes p-values, and wants to detect whether a nonzero effect exists (reject H_0).
3. The user stops the first time the p-value falls below α, up to a maximum run time of M.

Question: what always valid p-value processes deliver an efficient tradeoff between power and run-time, without advance knowledge of M?

Data model

For simplicity, we assume data generated from a $\mathcal{N}(\theta, 1)$ distribution, where θ is unknown.

We then consider testing:

$$
\begin{aligned}
H_0 &: \theta = 0 \\
H_1 &: \theta \neq 0
\end{aligned}
$$

More generally our theory holds for a single stream of data generated from a single parameter exponential family.

(We generalize to A/B tests — i.e., two streams — with binomial and normal data in the paper.)

The mSPRT

The mSPRT

Notation: Let $L_n(\theta, \theta_0; \overline{x}, n)$ be LR of θ against $\theta_0 = 0$, given n observations with sample mean $\overline{x}$.

Let $H \sim \mathcal{N}(0, \sigma^2)$, and consider:

$$\mathbf{L}_n = \int L_n(\theta, \theta_0; \overline{X}_n, n) dH(\theta).$$

Define:

$$S_\alpha = \inf \left\{ n : \mathbf{L}_n \geq \frac{1}{\alpha} \right\}.$$

This is the *mixture sequential probability ratio test* (mSPRT) due to Robbins and Siegmund.

Always valid p-values

It is straightforward to show using martingale techniques that the mSPRT is a sequential test, i.e., it controls Type I error at level α.

Always valid p-values

It is straightforward to show using martingale techniques that the mSPRT is a sequential test, i.e., it controls Type I error at level α.

In addition, the mSPRT has *power one*: it is guaranteed to detect any true effect eventually.

Always valid p-values

It is straightforward to show using martingale techniques that the mSPRT is a sequential test, i.e., it controls Type I error at level α.

In addition, the mSPRT has *power one*: it is guaranteed to detect any true effect eventually.

We use the corresponding always valid p-value process: p_n is the smallest α such that $S_\alpha \leq n$.

Thus for the mSPRT the always valid p-value is particularly simple:

$$p_n = \inf \left\{ \frac{1}{\mathbf{L}_k} \right\}.$$

Why it works

Under the null:

- Typical fluctuations of sample mean are of size $1/\sqrt{n}$, so any decision rule of the form:

  Reject if $|$*sample mean*$| > k/\sqrt{n}$

  is bound to eventually reject. This is what fixed horizon testing (e.g., z-test) will do.

Why it works

Under the null:

- Typical fluctuations of sample mean are of size $1/\sqrt{n}$, so any decision rule of the form:

  Reject if $|$*sample mean*$| > k/\sqrt{n}$

  is bound to eventually reject. This is what fixed horizon testing (e.g., z-test) will do.
- In fact, by law of the iterated logarithm, the boundary $\sqrt{2 \log \log n}/\sqrt{n}$ is crossed infinitely often.

Why it works

Under the null:

- Typical fluctuations of sample mean are of size $1/\sqrt{n}$, so any decision rule of the form:

  Reject if $|$*sample mean*$| > k/\sqrt{n}$

  is bound to eventually reject. This is what fixed horizon testing (e.g., z-test) will do.
- In fact, by law of the iterated logarithm, the boundary $\sqrt{2 \log \log n}/\sqrt{n}$ is crossed infinitely often.
- mSPRT leads to boundary of the form $C\sqrt{\log n}/\sqrt{n}$:
  - Goes to zero (power one);
  - But slowly enough (so Type I error can be controlled).

Efficiency

Although the mSPRT has power one, that is only asymptotically in the infinite data limit.

We show that the mSPRT trades off power and run-time efficiently, even when a user might abandon the test prematurely (at her personal maximum run-time M).

Efficiency

Given M, α, and θ, and an always valid p-value process, let:

- $R(\theta; M, \alpha) =$ expected run-time, and
- $q(\theta; M, \alpha) =$ false negative probability

when the true effect is θ, assuming the user stops at the lesser of M or the first time the p-value drops below α.

Efficiency

Given M, α, and θ, and an always valid p-value process, let:

- $R(\theta; M, \alpha) =$ expected run-time, and
- $q(\theta; M, \alpha) =$ false negative probability

when the true effect is θ, assuming the user stops at the lesser of M or the first time the p-value drops below α.

Definition

The p-value process is ϵ-*efficient* at (M, α) if for any other test with Type I error at most α, expected run length $\hat{R}(\theta)$, and false negative probability $\hat{q}(\theta)$, if:

$$(1+\epsilon)\hat{q}(\theta) \leq q(\theta; M, \alpha) \ \forall\, \theta \neq 0,$$

then

$$(1+\epsilon)R(\theta; M, \alpha) \leq \hat{R}(\theta) \ \forall\, \theta \neq 0.$$

Efficiency

Informally, ϵ-efficiency means that power cannot be appreciably increased without significantly inflating run-time (or vice versa).

Our efficiency result considers performance of the mSPRT in the "cheap data" limit, where $M \to \infty$.

Theorem

Consider a sequence of users with $M_k \to \infty$ and $\alpha_k \to 0$. Then the mSPRT leads to ϵ-efficient always valid p-values for all sufficiently large k, provided $\log(1/\alpha_k) = O(M_k)$.

Efficiency

Informally, ϵ-efficiency means that power cannot be appreciably increased without significantly inflating run-time (or vice versa).

Our efficiency result considers performance of the mSPRT in the "cheap data" limit, where $M \to \infty$.

Theorem

Consider a sequence of users with $M_k \to \infty$ *and* $\alpha_k \to 0$*. Then the mSPRT leads to* ϵ*-efficient always valid p-values for all sufficiently large* k*, provided* $\log(1/\alpha_k) = O(M_k)$*.*

(If the latter condition fails, then *any* sequential test controlling Type I error will have vanishingly small power.)

Interpretation

We interpret this result as follows:

- Users will have a range of risk preferences, encapsulated through α and M.

Interpretation

We interpret this result as follows:

- Users will have a range of risk preferences, encapsulated through α and M.
- We first control Type I error for all of them (always valid p-values).

Interpretation

We interpret this result as follows:

- Users will have a range of risk preferences, encapsulated through α and M.
- We first control Type I error for all of them (always valid p-values).
- Some users will be too conservative: α will be too small relative to M.
  For them, power will be small under any method.

Interpretation

We interpret this result as follows:

- Users will have a range of risk preferences, encapsulated through α and M.
- We first control Type I error for all of them (always valid p-values).
- Some users will be too conservative: α will be too small relative to M.
  For them, power will be small under any method.
- For the rest, α is reasonable relative to M. Among them, we focus on those with larger M.
  For these users, the mSPRT achieves an approximately efficient tradeoff between power and run-time, uniformly over θ.

Optimizing the mSPRT

How to choose the mixing distribution?

Since the mSPRT has power one with infinite data, we aim to optimize run-time.

In particular: assume effect θ is drawn from a prior G, and aim to minimize $\mathbb{E}[R(\theta; M, \alpha)]$.

Optimizing the mSPRT

We show that the optimal choice of H depends on the shape of G.

In the limit of $\alpha \to 0$, the optimal choice of mixing distribution in the mSPRT involves roughly *matching* the mixing variance σ^2 to the prior variance τ^2.

The constant of proportionality depends on $\log(1/\alpha)/M$, but (under reasonable values for prior) is relatively robust to changes in α or M.

Run length

No free lunch?

What do we give up in return for continuous monitoring?

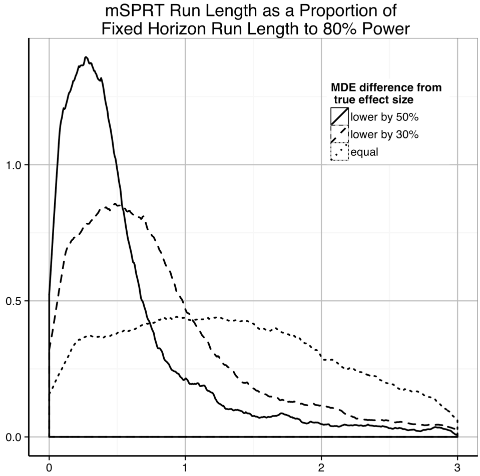
- If the effect size is known in advance,
  should only be better!
- In practice, we don't know the effect size in advance.

The test we designed does not assume knowledge of the effect size.

We compare our test to a fixed horizon test,
using data from Optimizely.

Run lengths on Optimizely

Our results show robustness to not knowing the effect size:



Run lengths: Interpretation

Our results show robustness to not knowing the effect size.
Intuition:

- Detecting an effect of size Δ takes a run length proportional to $1/\Delta^2$
- So the penalty for guessing wrong about δ is very high!
  - An MDE that is 2x too small $\Longrightarrow$ run length that is 4x too long

Run lengths: Theory

Suppose that effect θ is drawn from a normal distribution.

In an appropriate scaling regime where $\alpha \to 0$ and $N \to \infty$, we show that mSPRT at level α truncated to $\Theta(N)$ gives similar power as fixed horizon test of length N, but with detection time that is $o(N)$.

Run lengths: Theory

Suppose that effect θ is drawn from a normal distribution.

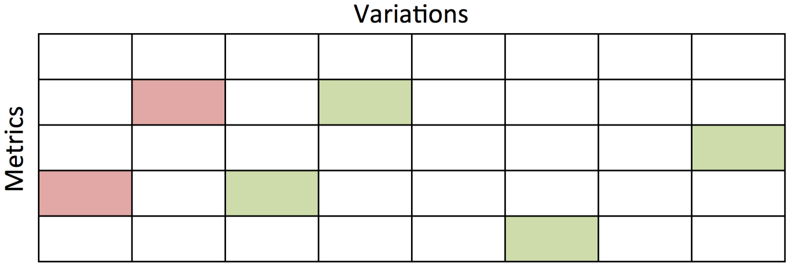
In an appropriate scaling regime where $\alpha \to 0$ and $N \to \infty$, we show that mSPRT at level α truncated to $\Theta(N)$ gives similar power as fixed horizon test of length N, but with detection time that is $o(N)$.

In other words: even users who *were* properly using fixed horizon p-values would prefer our approach, if effect size is uncertain.

Multiple testing

The multiple testing problem

Recall the typical dashboard of an A/B test:



Suppose each cell is an *independent* hypothesis test.
Note that if $\alpha = 0.1$,
expect 4 out of 40 to be significant by ***random chance***.

FWER and FDR

Suppose $K =$ number of hypotheses.

Can try to control:

- *Familywise error rate*: probability of making even one mistaken rejection
  - Standard approach to control: *Bonferroni correction*

FWER and FDR

Suppose $K =$ number of hypotheses.

Can try to control:

- *Familywise error rate*: probability of making even one mistaken rejection
  - Standard approach to control: *Bonferroni correction*
- *False discovery rate*: expected fraction of rejections that are mistaken
  - Less conservative
  - Standard approach to control: *Benjamini-Hochberg (BH) procedure*

FWER and FDR

Suppose $K =$ number of hypotheses.

Can try to control:

- *Familywise error rate*: probability of making even one mistaken rejection
  - Standard approach to control: *Bonferroni correction*
- *False discovery rate*: expected fraction of rejections that are mistaken
  - Less conservative
  - Standard approach to control: *Benjamini-Hochberg (BH) procedure*

Both use p-values as input. Since we generate p-values, can we apply these procedures?

Always validity, and FWER and FDR

The Bonferroni correction can be directly applied to
always valid p-values to provide always valid control of FWER.

Always validity, and FWER and FDR

The Bonferroni correction can be directly applied to
always valid p-values to provide always valid control of FWER.

More surprisingly, under reasonable assumptions,
always validity “commutes” with the BH procedure.

Always validity and FDR

We find a condition under which the BH procedure "commutes" with always validity.

Examples:

- Any stopping time that depends only on the sequence of the number of rejections made over time (e.g., the first time a fixed number of rejections is reached)
- The first time the p-value on a fixed hypothesis crosses a threshold

Always validity and FDR

In controlling FDR, what can go wrong?

- In general, the stopping time introduces dependence between the p-value processes.
- A result of Benjamini and Yekutieli shows:
  With arbitrary dependence among the hypotheses,
  the BH procedure at level α controls FDR at level $\alpha \ln K$.
- The same result then applies for always valid p-value processes.

Conclusions

Experimentation in the Internet age

Rapid innovation in
information & communication technology
has **democratized the scientific method**.

Our goal: “adapt” statistical methodology to
act in partnership with the user.

Additional results:

- Confidence intervals
- Adaptive allocation (bandits)

Optimizely Stats Engine

USING OPTIMIZELY

Statistics for the Internet Age: The Story Behind Optimizely's New Stats Engine

By Leonid Pekelis

- The ideas presented in this talk were released to all of Optimizely's customers on January 20, 2015
- Provides both always valid p-values
  and multiple testing corrections