Optimization vs Testing

Feel the Embrace of the Multi-armed Bandit

Brian Muller

bmuller@moonshinedev.co

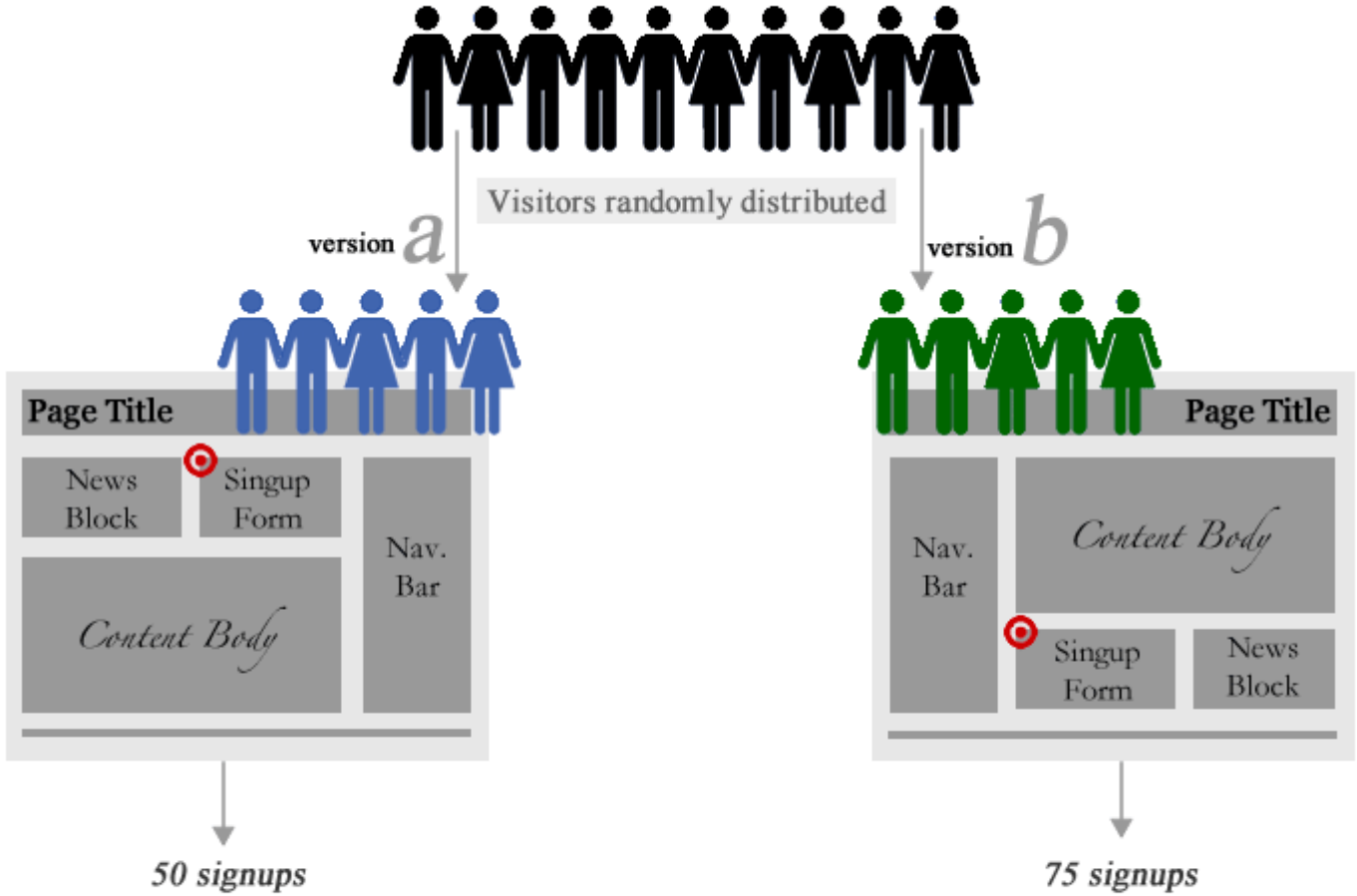
8 November 2012

Definition

A/B Testing

A/B Testing (aka, split testing) compares the effectiveness of two versions of a web page (content) to determine which has better "conversion" rate.

A/B Testing Diagram



Version B is better than version A

Complications

There are some potential complications, though:

1. How big of a difference is meaningful?
2. What should you do if the results are the same?
3. When should you stop?

Meaningful Deltas

Say we're testing two different email subject lines...

	Group A	Group B
Opened	100	135
Not Opened	200	200
Open / Not	0.5	0.675

Group B performed almost 20% better than Group A. Success?

Stopping Criteria

Result: p-value of 0.07, which means it's not unlikely[1] that we'd observe such a large difference *even if there isn't one*.

That is - there's a "decent chance" that the difference in the test doesn't mean anything.

So, run it longer?

[1] Note that "not unlikely" isn't the same thing as "probably."

Same Results

Say we're testing two different email subject lines, again...

	Group A	Group B
Opened	100	90
Not Opened	200	200
Open / Not	0.33	0.3

They're really close. Should we run the test for longer?

Solutions

To answer these questions correctly, we need to:

1. Determine the smallest delta that we care about
2. Determine our desired detection levels
3. Perform sample size calculations based on that delta and the baseline rate
4. Test for significance once we hit our desired size

Example

Let's say:

1. Our baseline conversion rate is 1%
2. We want to detect a 10% relative lift or larger (1% absolute to 11%)
3. We want a power (chance of detecting a difference when there is one) of 80%
4. We want a significance level of 5% (chance of detecting a difference that isn't there)

ABAnalyzer

To do this, we can use the ABAnalyzer gem.

```
# baseline conversion rate
baseline = 0.1

# detect a 10% relative lift (1% absolute to 11%) or larger
liftdetection = 0.11

# chance of detecting a difference that isn't there
sig = 0.05

# chance of detecting a difference when there is one
power = 0.8

# This will return 14751, which is the smallest number of
# people we need *in each group* - both test / control
ABAnalyzer.calculate_size(baseline, liftdetection, sig, power)
```

ABAnalyzer

We can also use the ABAnalyzer gem to test results (from the email campaign example).

```
groups = {
  :groupa => { :opened => 100, :notopened => 200 },
  :groupb => { :opened => 135, :notopened => 200 }
}

tester = ABAnalyzer::ABTest.new groups

# following will output 'Not different.'
puts (tester.different?) ? 'Different!' : 'Not different.'

# to see the actual p-value, which is 0.07
# (higher than 0.05 level of significance cutoff)
puts tester.gtest_p
```

A/B Testing

A/B Testing is

Reassess Goal

Reassess Goal

What if we could just maximize clicks at each step? Instead of evenly dividing audience, what if we just tried to divide the audience between the options to maximize clicks *for each page load*?

One Armed Bandit

Multi-armed Bandit

Multi-armed Bandit Definition

The multi-armed bandit problem describes a tradeoff at each stage. The player must choose between:

1. Exploration: Pulling an arm that hasn't been pulled before (or recently)
2. Exploitation: Pulling the arm that has performed the best so far

The goal is to maximizing the total reward over all at each step.

Benefits

If we view the problem this way, we get some benefits:

- Try risky options. Bad ones won't be shown often.
- Have as many alternatives as we want (domain can be exceptionally large).
- Conversions are maximized immediately - not after the test finishes.
- Alternatives can be added or removed at any time
- No one has to know what statistical power, significance, confidence intervals, etc. mean

Example Method: Round Robin

Round robin example.

```
# conversions = {
#   :firstchoice => 0.5,
#   :secondchoice => 0.4,
#   ...
# }

def pull_arm(conversions)
  # pick one randomly
  conversions.keys.sample
end
```

Example Method: Epsilon Greedy

Epsilon greedy example.

```
# conversions = {
#  :firstchoice => 0.5,
#  :secondchoice => 0.4,
#  ...
# }
# epsilon = 0.1

def pull_arm(conversions, epsilon)
  if rand > epsilon
    # get choice with max conversion 90% of the time
    conversions.max_by { |k, v| v }.first
  else
    # pick one randomly 10% of the time
    conversions.keys.sample
  end
end
```

Example Method: Epsilon-decreasing

Epsilon-decreasing example.

```
# conversions = {
#  :firstchoice => 0.5,
#  :secondchoice => 0.4,
#  ...
# }
# starttime = Time.now.to_i

def pull_arm(conversions, starttime)
  # 1 for first minute, then decreasing from there
  epsilon = [ 60.0 / (Time.now.to_i - starttime), 1 ].min
  if rand > epsilon
    conversions.max_by { |k, v| v }.first
  else
    conversions.keys.sample
  end
end
```

Bandit Gem

There's a (Rails) gem for this called bandit.

Example test configuration:

```
Bandit::Experiment.create(:click_test) { |exp|
  exp.alternatives = [ 20, 30, 40 ]
  exp.title = 'Click Test'
  exp.description = 'Purchase links with various sizes.'
}
```

Bandit Gem Usage

To get an alternative value in a view:

```
<%= bandit_choose :click_test %>
```

For instance, a link size:

```
<% style = 'font-size: #{bandit_choose(:click_test)}px;' %>
<%= link_to 'new purchase', new_purchase_path, :style => style %>
```

Bandit Gem Conversion Tracking

To track a conversion in your controller:

```
bandit_convert! :click_test
```

You can also request a choice in the controller:

```
redirect_to bandit_choose(:some_url_test)
```

Dashboard

There's a dashboard:

Bandit Dashboard

index

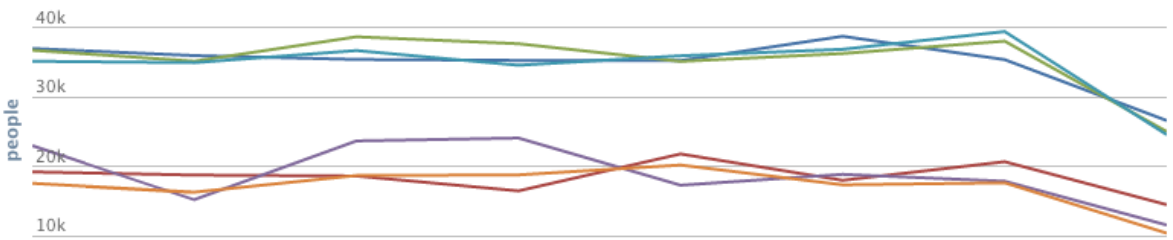
Click Test

A test of clicks on purchase page with varying link sizes.

Alternative	Participants	Conversions	Conversion Rate
20	279063	147206	52.75 %
30	282141	150734	53.43 %
40	277614	136065	49.01 %

Click Test
participant / conversion totals

Zoom 1m 3m 6m YTD 1y All From: Dec 1, 2012 To: Dec 8, 2012

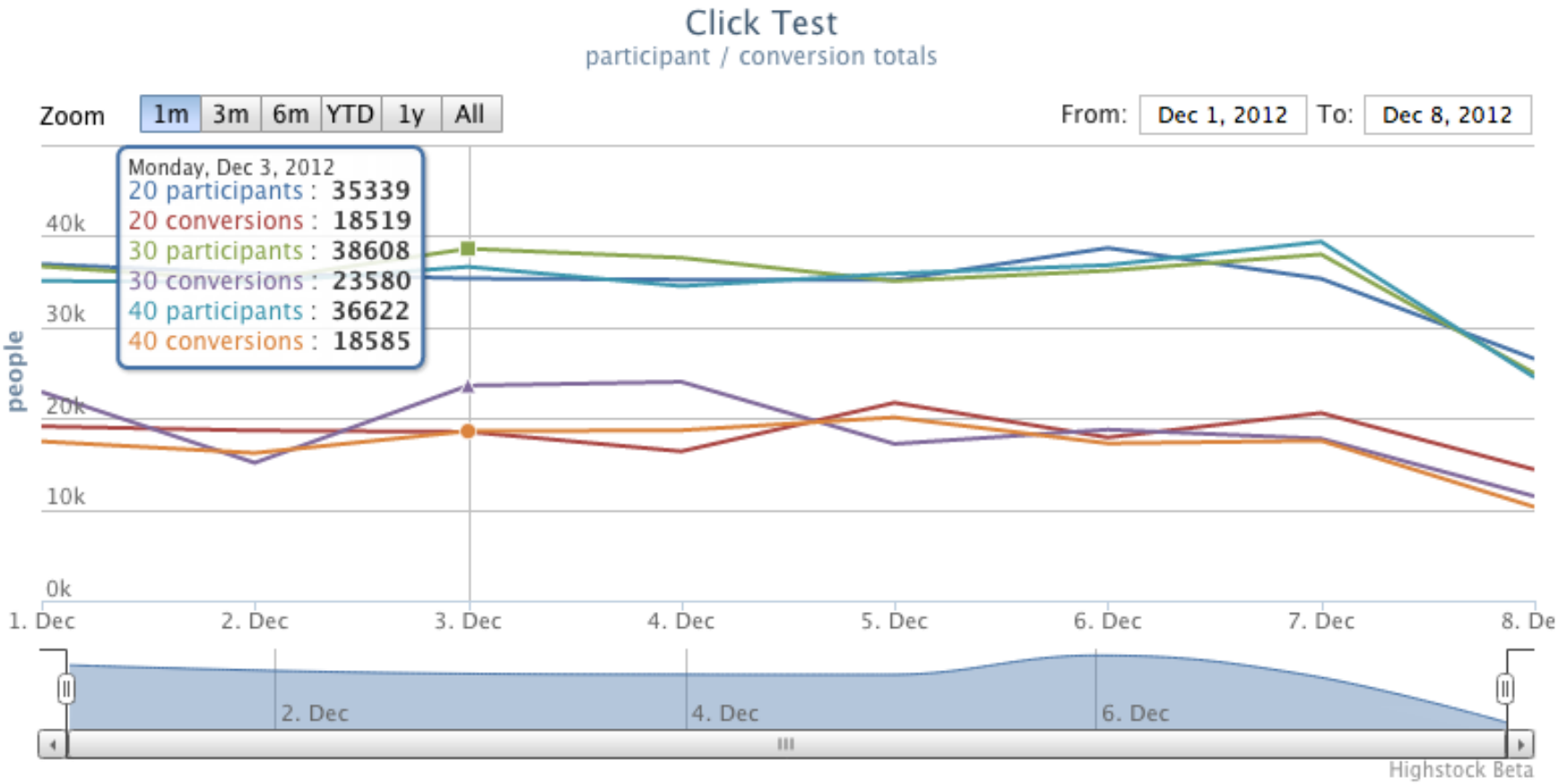


Experiments

Click Test

Results

Graphed results:



Testing Usage

A/B Testing is good when:

- you know that the best option is permanent
- you know that the best option is global
- there are limited alternatives

Multi-armed Usage

Multi-armed bandit optimization is good when:

- conversion rates may change over time
- you may have "risky" alternatives
- there are many alternatives (could be an infinite number)
- you may be adding/removing alternatives regularly

Requisite Plug

At the Moonshine Dev Co, we're working on a commercial content optimization service at opbandit.com.

Questions

Questions?