You Won't Believe How We Optimize Our Headlines

DataEngConf 2017

Lucy X Wang
BuzzFeed

Optimizing A Headline Optimizer

DataEngConf 2017

Lucy X Wang
BuzzFeed

Building an Optimizer

$$\frac{\textit{successes}}{\textit{trials}}$$

DataEngConf 2017

Lucy X Wang
BuzzFeed

BuzzFeed

Our headlines and thumbnail images span a wide range of post types

What Colors Are This Dress?

"A Honeypot For Assholes": Inside Twitter's 10-Year Failure To Stop Harassment

These Reports Allege Trump Has Deep Ties To Russia

The *Optimizer*

FlexPro: a BuzzFeed service that writers use to choose the best headline and thumbnail combination for an article post

Here are the optimization results for your post! Here are the top three headline + thumbnail combinations:

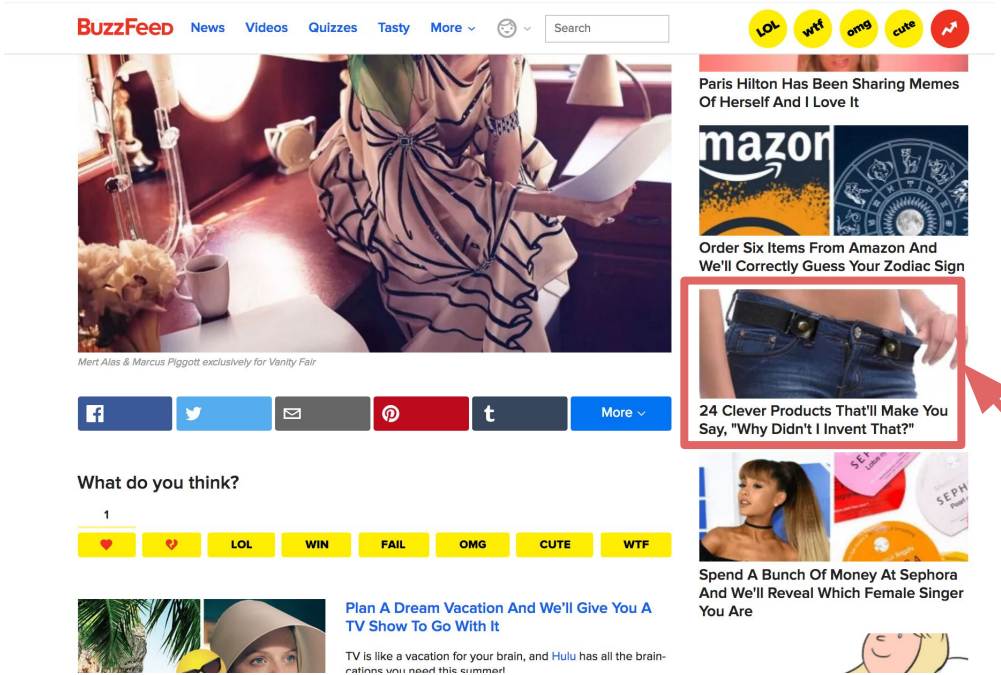
1 1.84x *performance* 🏆
24 Clever Products That'll Make You Say, "Why Didn't I Invent That?"

2 1.76x *performance*
24 Clever Products That'll Make You Say, "Why Didn't I Invent That?"

3 1.51x *performance*
24 Ridiculously Clever Products You'll Wish You'd Invented Yourself

Top 3 winning variants for a test

The *Optimizer*



- Tests all the submitted headline x thumbnail combinations (variants) live on buzzfeed.com
- Measures clicks and impressions on every variant
- Selects the winning combination, which becomes the default headline and thumbnail for the article

During test, each variant of the post is simultaneously shown to a distinct subset of users on the site

some *press*

“BuzzFeed also has tools like a headline optimizer. It can take a few different headline and thumbnail image configurations and test them in real time as a story goes live, then spit back the one that is most effective.”

Inside the Buzz-Fueled Media Startups Battling for Your Attention, WIRED, 2014

The OG FlexPro

- Version 1 tests the variants live on the site using Multi-Armed Bandits
- Variants with higher CTR get increased exposure on the site in a greedy fashion
- Eventually, a winning variant is selected, when its CTR is deemed highest by a statistically significant margin

The Problem

Need for Speed

Social platform performance had become a product priority

The fastest winner selection algorithm allows us to distribute the optimized version of the article on social platforms. If too slow, we publish the non-optimized version.

1 1.0x *performance (Original headline and thumb)*

Kathy Griffin Accused Andy Cohen Of Offering Her Cocaine In A Blistering Video

2 0.98x *performance*

Kathy Griffin Drags TMZ, Claims Andy Cohen Offered Her Cocaine On "WWHL"

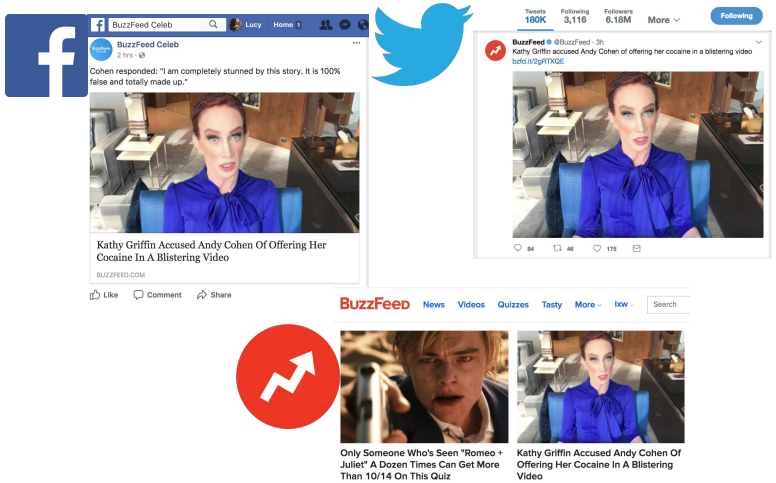
3 0.98x *performance*

Kathy Griffin Drags TMZ, Claims Andy Cohen Offered Her Cocaine On "WWHL"

test variants

Kathy Griffin Accused Andy Cohen Of Offering Her Cocaine In A Blistering Video

select winner



disseminate winner

Out with the Old

A new FlexPro algorithm was needed to select experiment winners with statistical rigor and speed

- Experiments taking too long to complete with the legacy algorithm (>12 hours)
- Promptly publishing the article on social platforms (Facebook) requires optimal headline and thumbnail output ASAP
- Had critical dependencies on other services that were getting decommissioned

The Algorithm

Methodology

Given the new prioritization on speed of variant testing:
Try a new algorithm to get faster results

Old algorithm: Multi-Armed Bandit	**New algorithm: Bayesian A/B Testing**
➢ Ensures that higher performing variants get increased exposure on site ➢ Significance will take longer to get established ➢ Maximizes the clicks received on the site	➢ Gives max impressions to every variant, including worse-performing variants ➢ Minimizes the duration of each test ➢ Gives intuitive results e.g. probability that A is the best variant, and expected CTR loss

Bayesian A/B Test Approach

1. Fit the posterior probability density distributions of each variant's CTR using a **beta distribution**:

   $P(CTR \mid clicks, impressions) \sim \boldsymbol{B}(\alpha = clicks, \beta = impressions - clicks)$

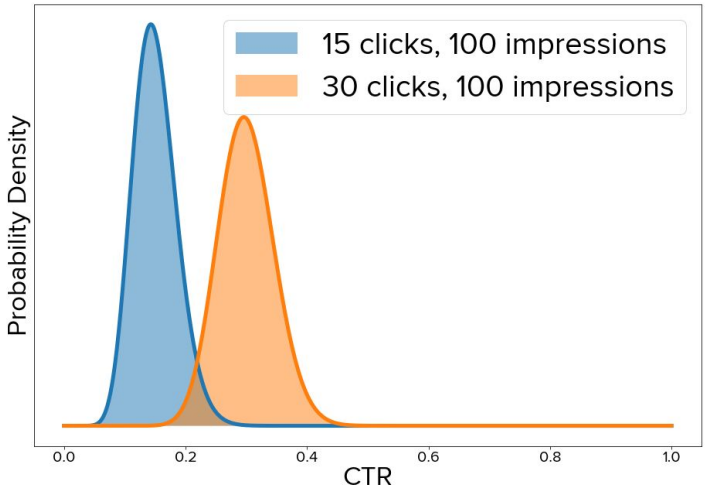
2. Calculate the probability that variant A is better than B (and C, D, ...) based on these pdfs

3. Use these probabilities to calculate expected loss for each variant (e.g. how many clicks **can** I **possibly lose** if I choose this variant as winner?) All choices come with a potential risk.

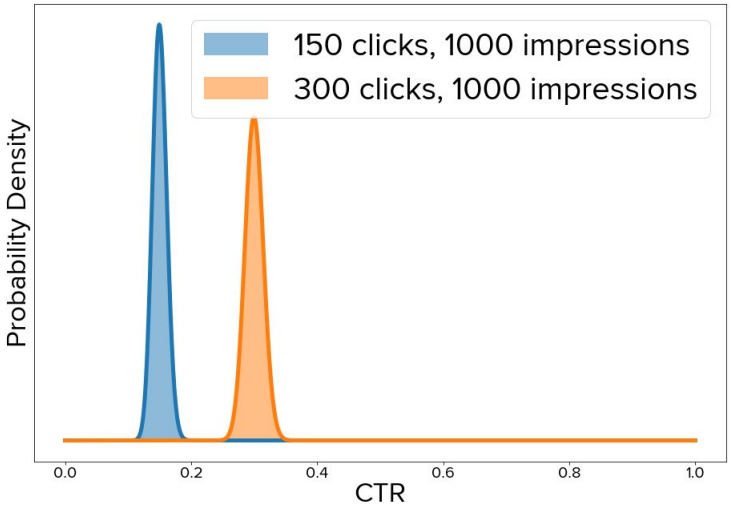
4. Don't decide on a winner until you can guarantee its expected loss falls below a "**threshold of caring"** defined in advance

Bayesian A/B Test Approach

- Winner was already obvious with less trials(left)
- Even though more trials helps (right)
- Can resolve ASAP with less trials (left)



trials x 10



Aside:

Closed Form Probability Formulas.... FML

Must calculate P(variant A > variant B)

... but deriving a closed form solution for this AND translating it to code is painful

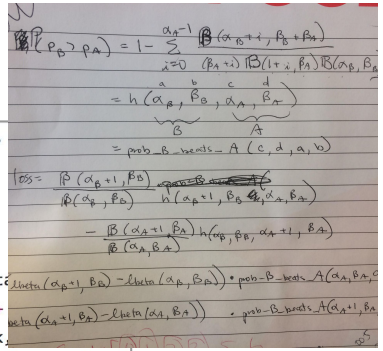
.... even trickier when number of variants > 2

$$
\begin{aligned}
\Pr(\lambda_1 > \lambda_2) &= \int_0^\infty e^{-\beta_1\lambda_2}\left(\sum_{k=0}^{\alpha_1-1}\frac{(\beta_1\lambda_2)^k}{\Gamma(k+1)}\right)\frac{\beta_2^{\alpha_2}\lambda_2^{\alpha_2-1}e^{-\beta_2\lambda_2}}{\Gamma(\alpha_2)}d\lambda_2 \\
&= \sum_{k=0}^{\alpha_1-1}\int_0^\infty \frac{e^{-\beta_1\lambda_2}(\beta_1\lambda_2)^k}{\Gamma(k+1)}\frac{\beta_2^{\alpha_2}\lambda_2^{\alpha_2-1}e^{-\beta_2\lambda_2}}{\Gamma(\alpha_2)}d\lambda_2 \\
&= \sum_{k=0}^{\alpha_1-1}\frac{\beta_1^k\beta_2^{\alpha_2}}{\Gamma(k+1)\Gamma(\alpha_2)}\int_0^\infty e^{-(\beta_1+\beta_2)\lambda_2}\lambda_2^{k+\alpha_2-1}d\lambda_2 \\
&= \sum_{k=0}^{\alpha_1-1}\frac{\beta_1^k\beta_2^{\alpha_2}}{\Gamma(k+1)\Gamma(\alpha_2)}(\beta_1+\beta_2)^{-(k+\alpha_2)}\Gamma(k+\alpha_2) \\
&= \sum_{k=0}^{\alpha_1-1}\beta_1^k\beta_2^{\alpha_2}(\beta_1+\beta_2)^{-(k+\alpha_2)}\frac{\Gamma(k+\alpha_2)}{\Gamma(k+1)\Gamma(\alpha_2)}\frac{k+\alpha_2}{k+\alpha_2}
\end{aligned}
$$

$$
\Pr(\lambda_1 > \lambda_2) = \sum_{k=0}^{\alpha_1-1}\beta_1^k\beta_2^{\alpha_2}(\beta_1+\beta_2)^{-(k+\alpha_2)}\frac{\Gamma(k+\alpha_2+1)}{\Gamma(k+1)\Gamma(\alpha_2)}\frac{1}{k+\alpha_2}
$$

```
f probability_D_beats_A_B_C(alpha_A, beta_A, alpha_B, beta_B, alpha_C, beta_C, alpha_D,
    total = 0
    for i in range(alpha_A):
        for j in range(alpha_B):
            for k in range(alpha_C):
                total += np.exp(lbeta(alpha_D + i + k, beta_A + beta_B + beta_C + beta
                                      np.log(beta_B + j) - np.log(beta_C +
                                      lbeta(1 + i + beta_A) - lbeta(1 + j, beta_B) - lbeta(1 + k,
                                      lbeta(alpha_D, beta_D) )
    return 1 - probability_B_beats_A(alpha_D, beta_D, alpha_A, beta_A) - \
        probability_B_beats_A(alpha_D, beta_D, alpha_B, beta_B) - \
            probability_B_beats_A(alpha_D, beta_D, alpha_C, beta_C) + total
```

wtf



Using Monte Carlo Instead

Simple Idea: P(variant A > variant B) can be approximated by the number of times a random draw from A's CTR distribution is > a random draw from B's CTR distribution

Repeat this 1000x (or more for better precision)

```
In [40]: A = beta(10, 100-10, 10000) # 10 successes, 100 trials
         B = beta(15, 100-15, 10000) # 15 successes, 100 trials
         np.greater(A,B).mean() # P(A > B)

Out[40]: 0.13289999999999999
```

Simulating the Expected Losses

Every choice comes with a risk.
Calculate the expected loss of choosing **variant A** as the winner:

1. Randomly draw from every variant's CTR distribution.
2. If variant A's CTR is the highest:
   expected loss = 0
3. If a different variant's CTR is highest:
   expected loss = max variant CTR - variant A CTR.
4. Repeat for 1000 random draws.
5. Average the losses across the 1000 draws.

The output is the loss in CTR you can expect from choosing variant A over all other variants.

How Much Loss Is Acceptable?

- Only choose a variant as winner when its expected CTR loss falls below a pre-defined **threshold of caring:** *the potential loss in CTR that you are willing to risk*
- Example values for : 0.01%, 0.005%, 0.00001%. Real intuitive!
- If it does not fall below this threshold, keep testing.

Resolving Inconclusive Tests

- Major motivation for version 2 is to keep experiments fast!
- We impose a hard, self-defined limit on the number of impressions a variant can receive: the impression_limit
- If no winner is statistically significant by the time the impression_limit is reached: default to writer's discretion.
- But wait...

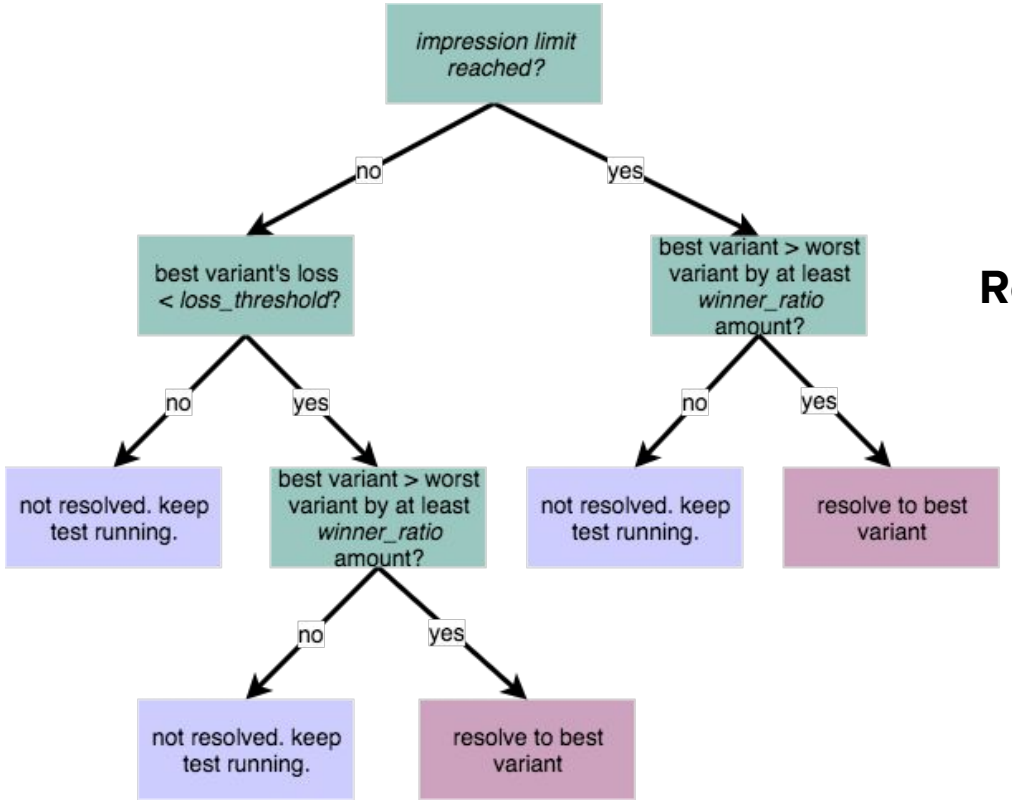
What about Ties?

- The method I started out with will only identify if there is a clear winner

- What if there is only a clear loser?!

- **Idea**: Choose either A or B randomly so long as the choice outperforms the worst variant (C) by a *certain ratio*. That way, the clear losers are at least thrown out.

Final Product



Resolve time: 1 day -> 1.5 hours!

Measuring Impact

Evaluation Goal

We needed to quantify FlexPro version 2's impact on post views

1. Relative to not using an optimizer at all, AND
2. Relative to version 1's impact

Hypothesis

1. Version 2 (Bayesian A/B Testing) will perform best in social platform views
2. Version 1 (Multi Armed Bandit) will perform best in onsite views

Can’t A/B Test ¯_(ツ)_/¯

A proper A/B test was out of the question.

1. A post can only stick with one headline and thumbnail when shared on social platforms. Therefore we cannot compare the outputs of two algorithms in a controlled setting

2. Version 1 had to be deprecated for other reasons; could not resurrect

Naive Approach

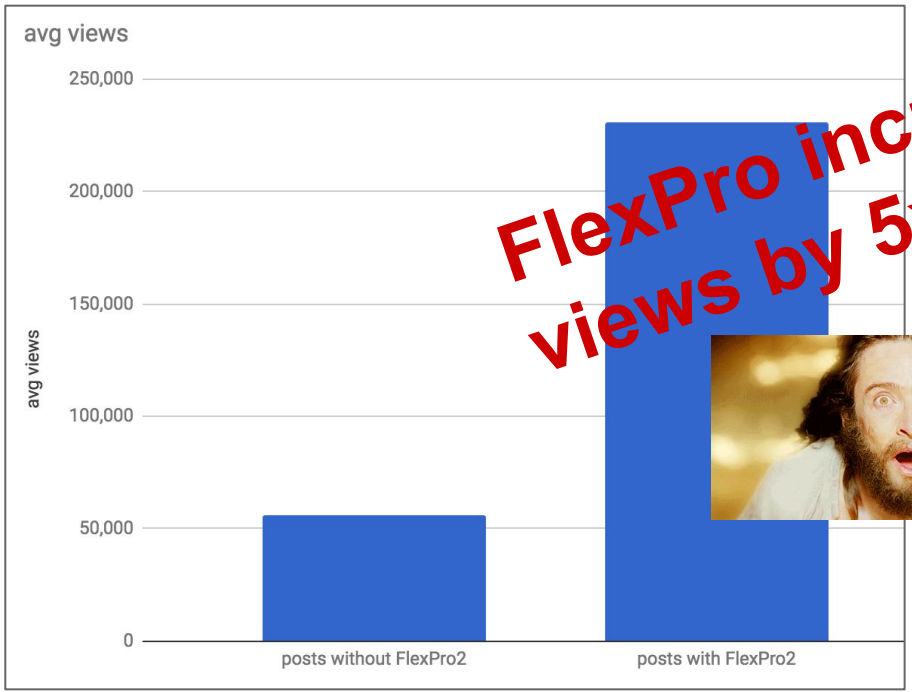
All posts with FlexPro on are in the **test group.**

All posts with FlexPro off are in the **control group.**

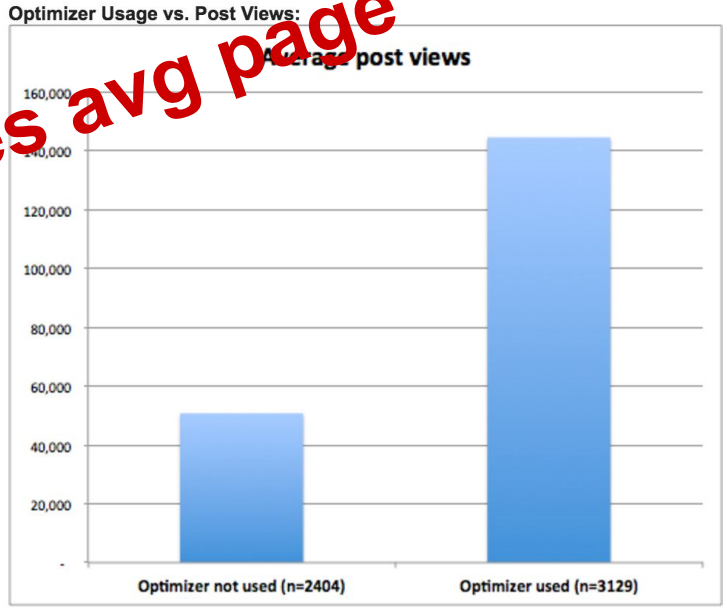
Result:

- FlexPro off posts: average of **56K** views
- FlexPro on posts: average of **231K** views

Naive Approach



FlexPro increases avg page views by 5x!



Using the Optimizer = More Post Views

Communication from 2015 about v1

A Causal Approach

Problem: FlexPro usage may correlate with other factors e.g. the post's author, vertical, etc.

Data: Each data point is a post with features:

flexpro_on: Was FlexPro used?
vertical: The post's category e.g. News, Quiz, etc.
author: The post's author

Idea: Use **propensity matching** to group these posts into *pseudo* ***treatment*** *and* ***control*** groups, where FlexPro on is a *treatment*. Treatment group members should *behave* similarly to their control group counterparts.

Measurement: What is the avg # views for treatment group vs control group?

Propensity Matching

- To measure the efficacy of a drug, you want to ensure that your treatment subjects and your control subjects have equal likelihood of going after the drug.
- Posts have different propensities for using FlexPro, and that can be based on the author, vertical, etc. of the post.
- Fit **logistic regression** Model:

  flexpro_on ~ author + vertical

- **Propensity scores = model's class probabilities**

  P(flexpro_on = 1 | author='Matt Perpetua', vertical='Quiz')

- For every member of treatment group (flexpro on), add a member to control group (flexpro off) with nearest propensity

Estimating Treatment Effect

- Fit a **linear regression model** on the new dataset to get fitted β values

$$\#\text{views} = \beta_1 \text{flexpro\_on} + \beta_2 \text{author} + \beta_3 \text{vertical}$$

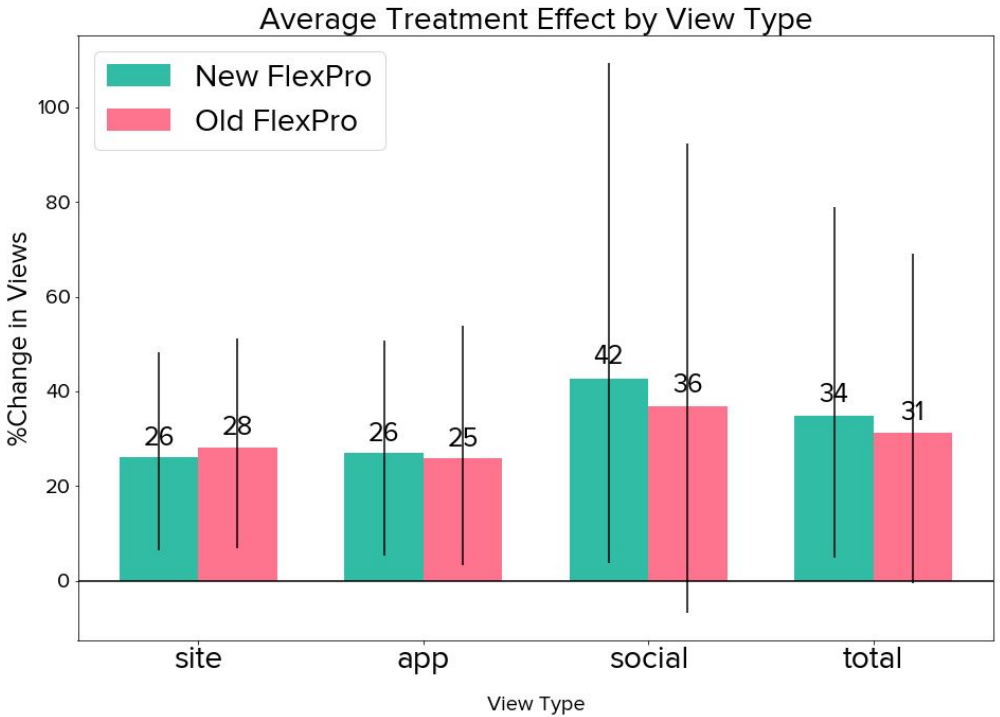
β_1 = the average treatment effect (ATE) of flexpro

- Repeated this whole process on n bootstrapped samples to generate confidence intervals for average treatment effect of flexpro

Conclusion

LARGE error bars

Effect on views is positive for both v1 and v2.

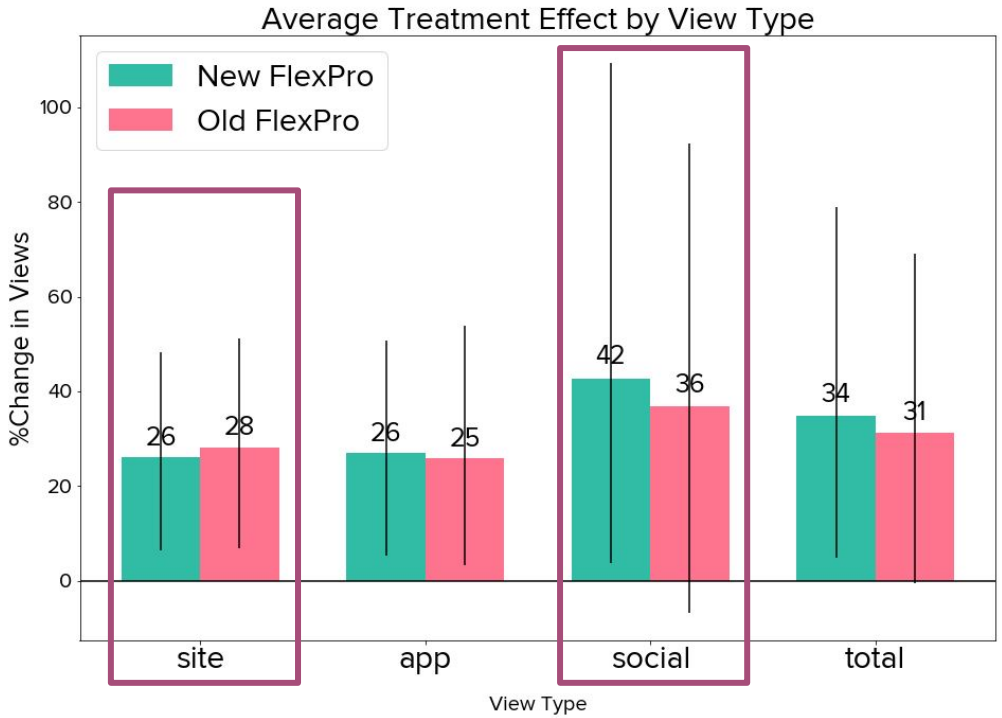


Conclusion

As hypothesized,

- Bayesian A/B Testing better for speed and Social platform views
- Multi Armed Bandit better for Site views

No 5x improvement, but will accept 1.35x



Thank you!

Psst -- we’re hiring!

lucy.wang@buzzfeed.com