Introduction to A/B testing

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON

Ryan Grossman
Data Scientist, EDO

Overview

- Introduction to A/B testing
- How to design an experiment
- Understand the logic behind A/B testing
- Analyze the results of a test

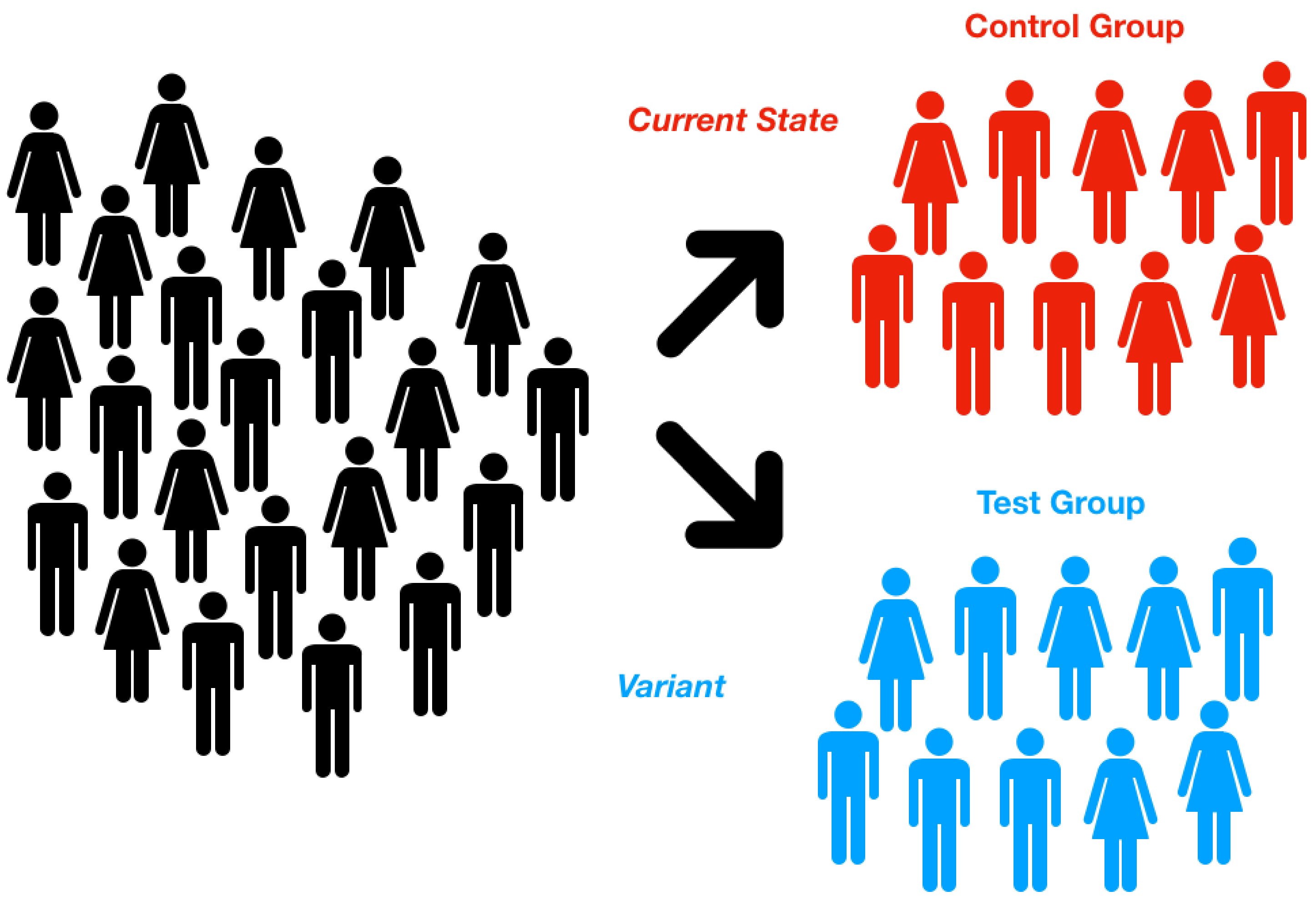
A/B test: an experiment where you...

- Test two or more variants against each other
- to evaluate which one performs "best",
- in the context of a randomized experiment

Control and treatment groups

Testing two or more ideas against each other:

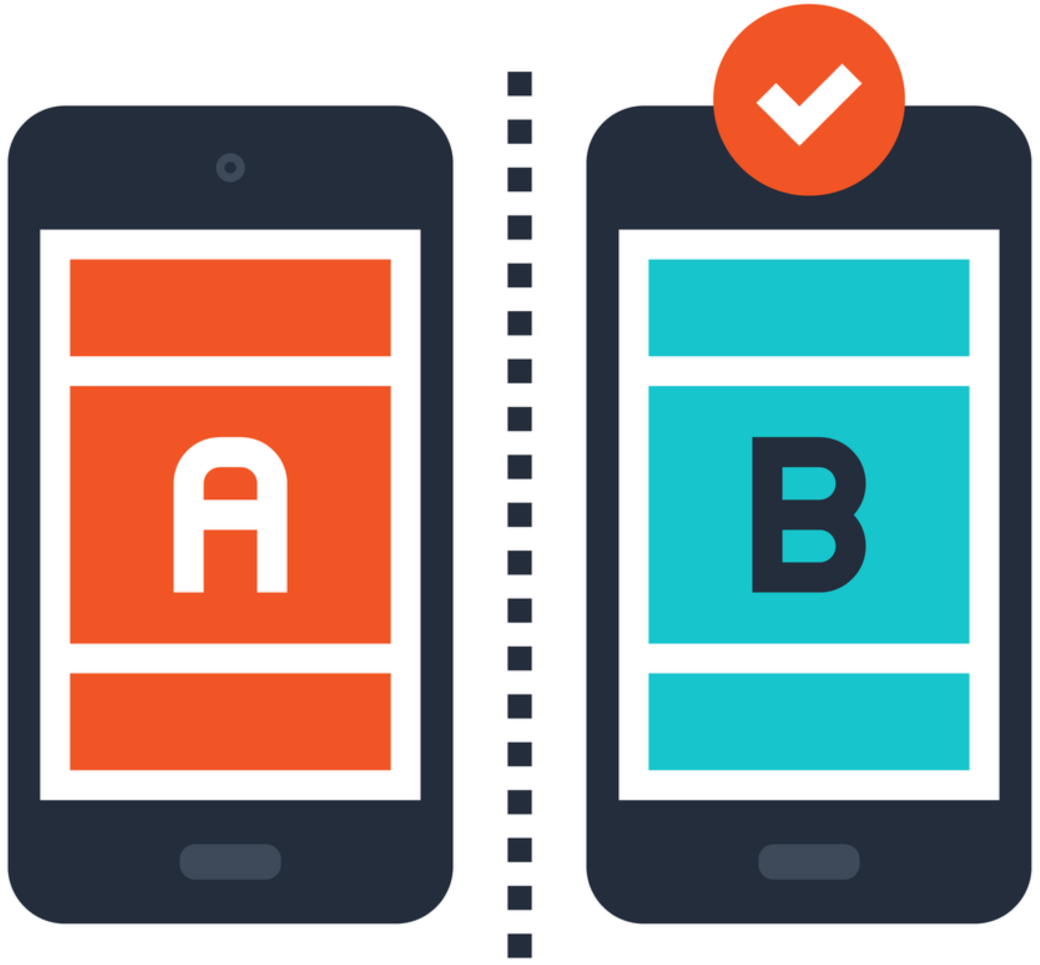
- **Control:** The current state of your product
- **Treatment(s):** The variant(s) that you want to test



A/B Test - improving our app paywall

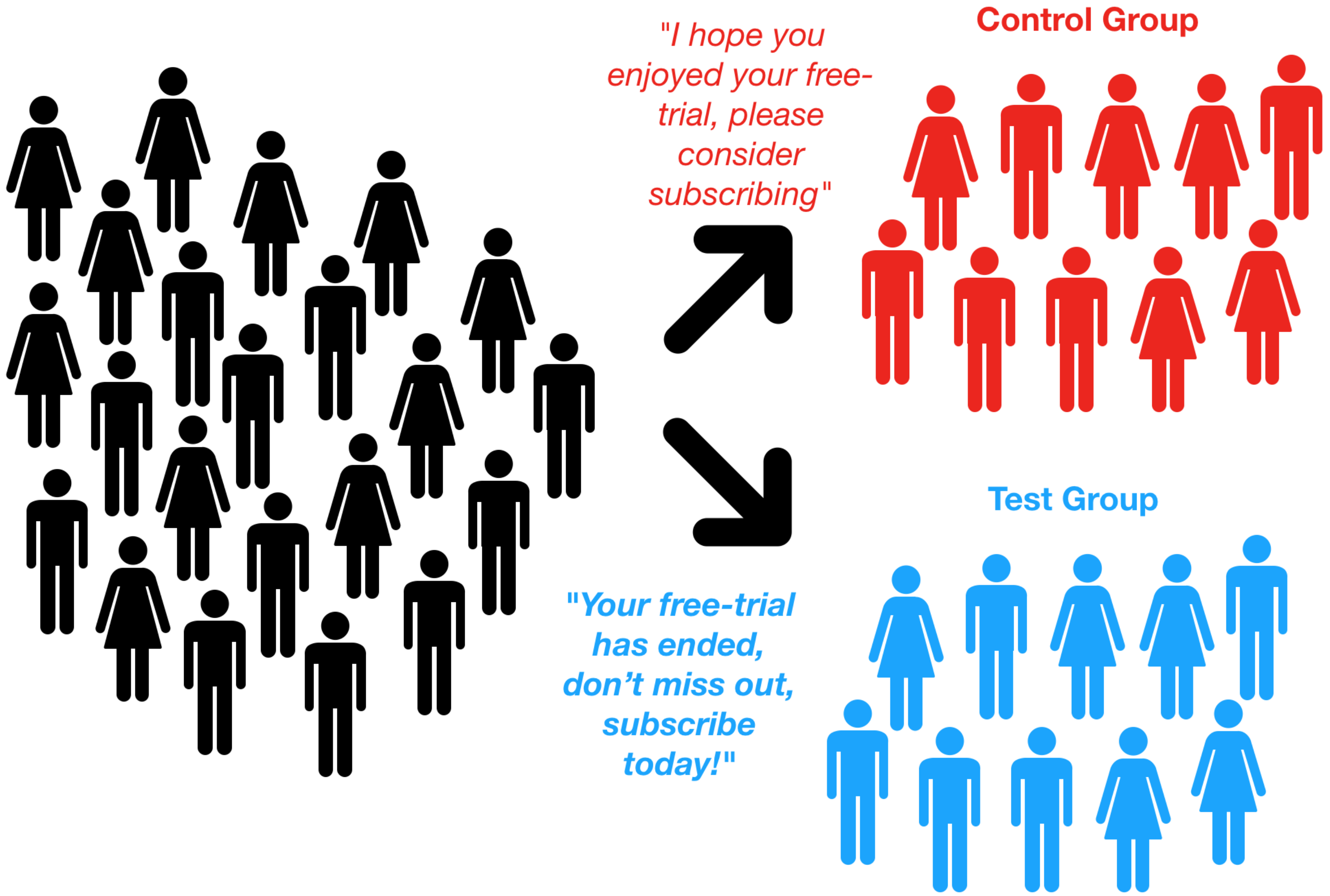
Question: Which paywall has a higher conversion rate?

- **Current Paywall:** *"I hope you enjoyed your free-trial, please consider subscribing"* (control)
- **Proposed Paywall:** *“Your free-trial has ended, don’t miss out, subscribe today!”* (treatment)



A/B testing process

- Randomly subset the users and show one set the control and one the treatment
- Monitor the conversion rates of each group to see which is better



The importance of randomness

- Random assignment helps to...
  - isolate the impact of the change made
  - reduce the potential impact of confounding variables
- Using an assignment criteria may introduce confounders

A/B testing flexibility

- A/B testing can be use to...
  - improve sales within a mobile application
  - increase user interactions with a website
  - identify the impact of a medical treatment
  - optimize an assembly lines efficiency
  - and many more amazing things!

Good problems for A/B testing

- Users are impacted individually
- Testing changes that can directly impact their behavior

Bad problems for A/B testing

- Cases with network effects among users
  - Challenging to segment the users into groups
  - Difficult to untangle the impact of the test

Let's practice!

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON

Initial A/B test design

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON

Ryan Grossman
Data Scientist, EDO

Increasing our app's revenue with A/B testing

Specific Goals:

- Test change to our consumable purchase paywall to...
- Increase revenue by increasing the purchase rate

General Concepts:

- A/B testing techniques transfer across a variety of contexts
- Keep in mind how you would apply these techniques

Paywall views & Demographics data

```
demographics_data = pd.read_csv('user_demographics.csv')
demographics_data.head(n=2)
```

```
         uid             reg_date device gender country  age
0  5.277e+07  2018-03-07T00:00:00Z    and      F     FRA   27
1  8.434e+07  2017-09-22T00:00:00Z    iOS      F     TUR   22
```

```
paywall_views = pd.read_csv('paywall_views.csv')
paywall_views.head(n=2)
```

```
        uid                      date  purchase  sku  price
0  32209877 2016-12-04 14:20:49+00:00         0  NaN    NaN
1  32209877 2016-12-05 22:17:12+00:00         0  NaN    NaN
```

Chapter 3 goals

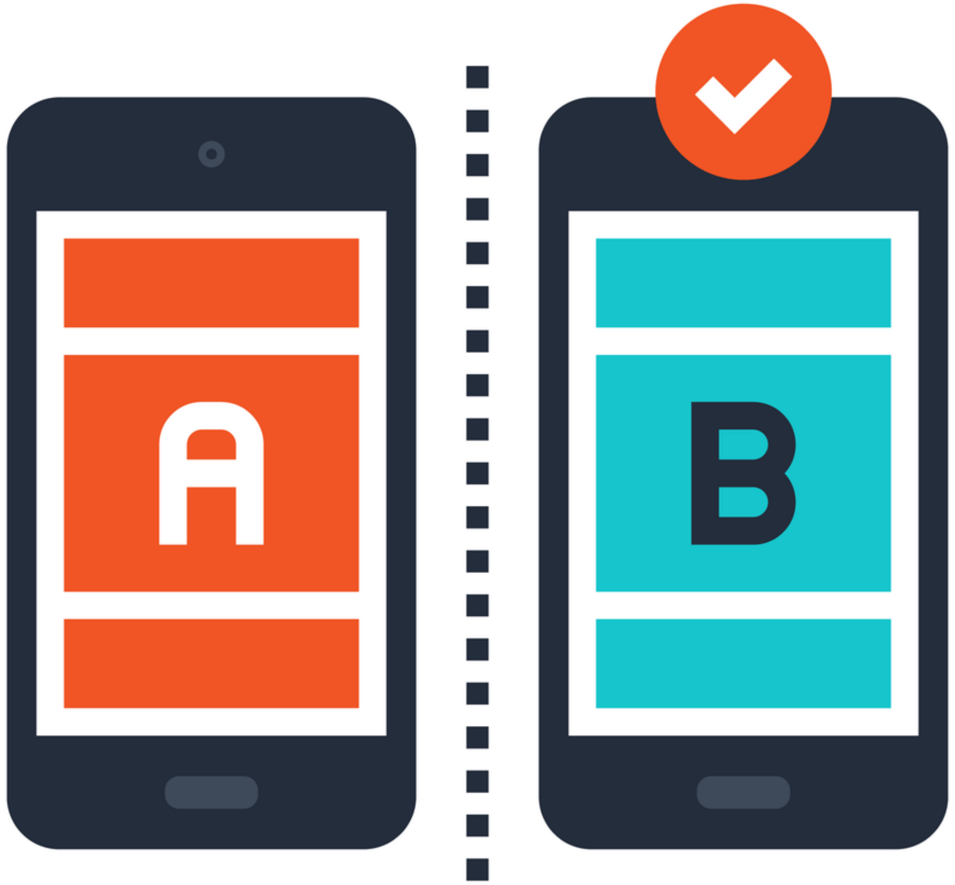
- Introduce the foundations of A/B testing
- Walk through the code need to apply these concepts

Response variable

- The quantity used to measure the impact of your change
- Should either be a KPI or directly related to a KPI
- The easier to measure the better

Factors & variants

- **Factors:** The type of variable you are changing
  - *The paywall color*
- **Variants:** Particular changes you are testing
  - *A red versus blue paywall*



Experimental unit of our test

- The smallest unit you are measuring the change over
- Individual users make a convenient experimental unit

Calculating experimental units

```python
# Join our paywall views to the user demographics
purchase_data = demographics_data.merge(
    paywall_views, how='left', on=['uid'])

# Find the total purchases for each user
total_purchases = purchase_data.groupby(
    by=['uid'], as_index=False).purchase.sum()
# Find the mean number of purchases per user
total_purchases.purchase.mean()
```

```
3.15
```

Calculating experimental units

```python
# Find the minimum number of purchases made by a user
# over the period
total_purchases.purchase.min()
```

```
0.0
```

```python
# Find the maximum number of purchases made by a user
# over the period
total_purchases.purchase.max()
```

```
17.0
```

Experimental unit of our test

User-days: User interactions on a given day

- More convenient than users by itself
- Not required to track user's actions across time
- Can treat simpler actions as responses to the test

Calculating user-days

```
# Group our data by users and days, then find the total purchases
total_purchases = purchase_data.groupby(
    by=['uid', 'date'], as_index=False)).purchase.sum()

# Calcualte summary statistics across user-days
total_purchases.purchase.mean()
total_purchases.purchase.min()
total_purchases.purchase.max()
```

```
0.0346
0.0
3.0
```

Randomness of experimental units

- Best to randomize by individuals regardless of our experimental unit
- Otherwise users can have inconsistent experience
- This can impact the tests results

Designing your A/B test

- Good to understand the qualities of your metrics and experimental units
- Important to build intuition about your users and data overall

Let's practice!

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON

Preparing to run an A/B test

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON

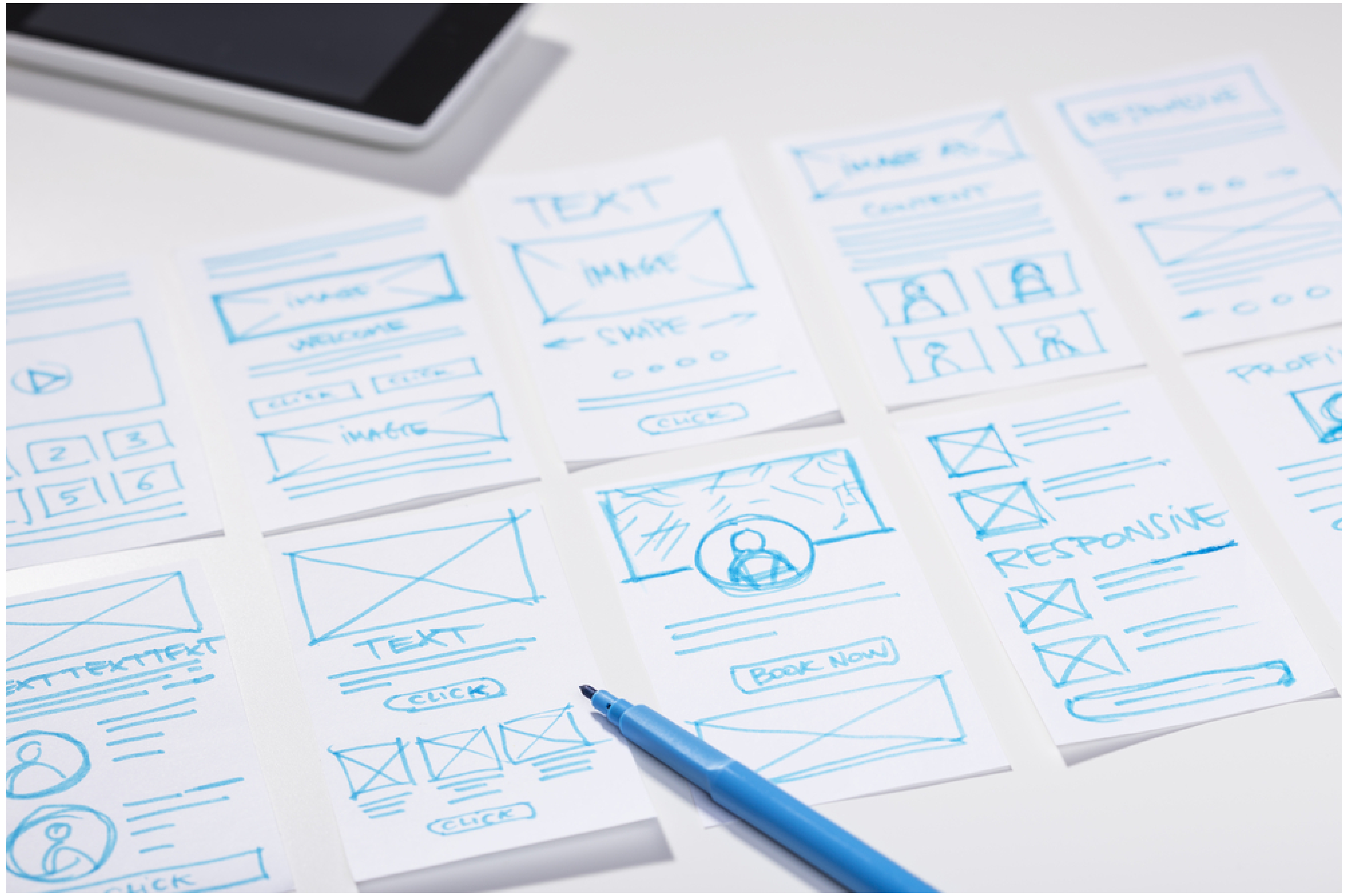
Ryan Grossman
Data Scientist, EDO

A/B testing example - paywall variants

- **Paywall Text: Test & Control**
  - **Current Paywall:** *"I hope you are enjoying the relaxing benefits of our app. Consider making a purchase."*
  - **Proposed Paywall** *Don't miss out! Try one of our new products!*
- **Questions**
  - Will updating the paywall text impact our revenue?
  - How do our three different consumable prices impact this?

Considerations in test design

1. Can our test be run well in practice?
2. Will we be able to derive meaningful results from it?



Test sensitivity

- **First question:** What size of impact is meaningful to detect
  - 1%...?
  - 20%...?
- Smaller changes = more difficult to detect
  - can be hidden by randomness
- **Sensitivity:** The minimum level of change we want to be able to detect in our test
  - Evaluate different sensitivity values

Revenue per user

```
# Join our demographics and purchase data
purchase_data = demographics_data.merge(
    paywall_views,how='left', on=['uid'])

# Find the total revenue per user over the period
total_revenue = purchase_data.groupby(by=['uid'], as_index=False).price.sum()
total_revenue.price = np.where(
    np.isnan(total_revenue.price), 0, total_revenue.price)

# Calculate the average revenue per user
avg_revenue = total_revenue.price.mean()
print(avg_revenue)
```

```
16.161
```

Evaluating different sensitivities

```
avg_revenue * 1.01    # 1% lift in revenue per user
```

```
16.322839545454478
```

```
# Most reasonable option
avg_revenue * 1.1     # 10% lift in revenue per user
```

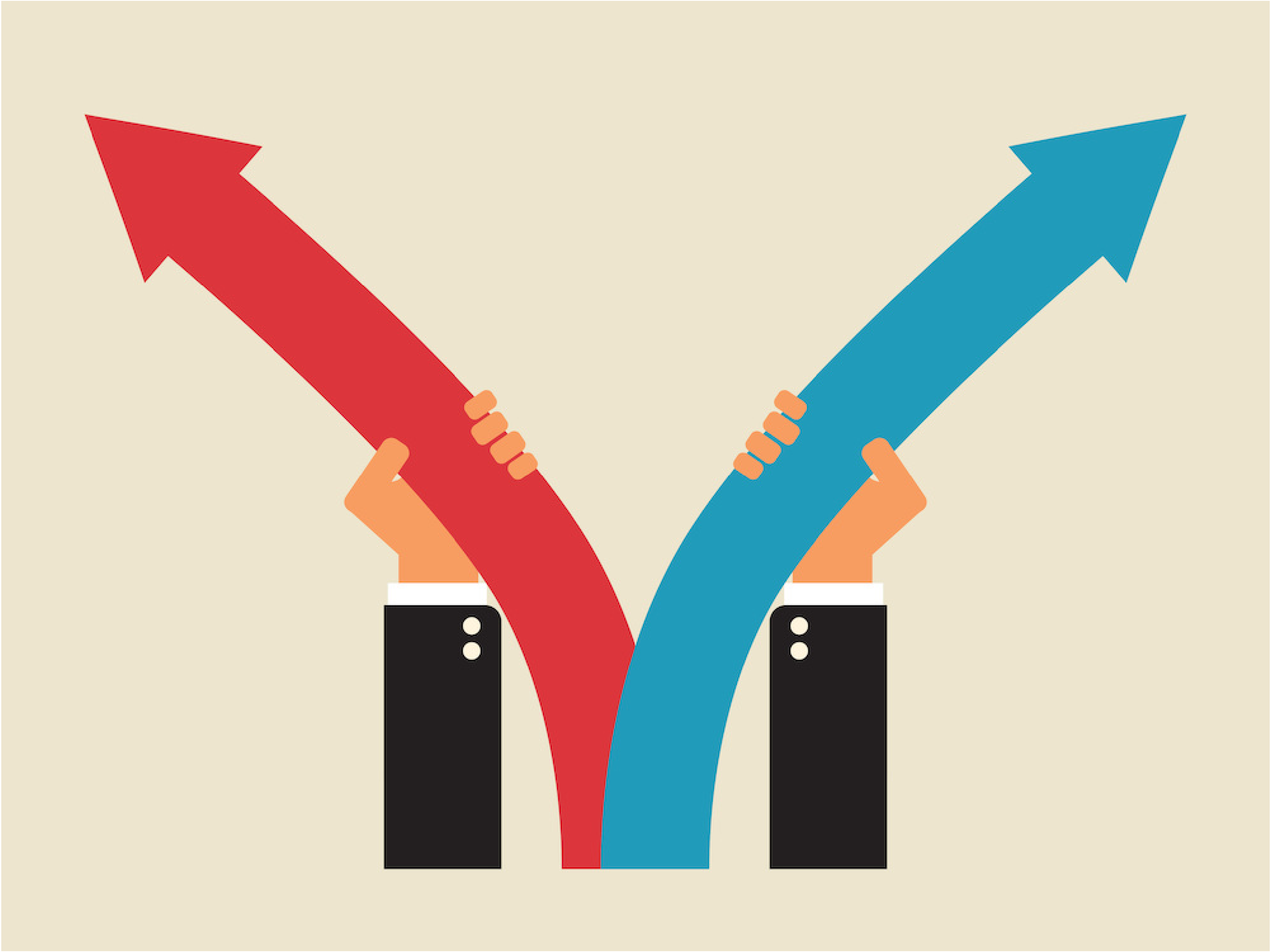
```
17.77
```

```
avg_revenue * 1.2     # 20% lift in revenue per user
```

```
19.393
```

Data variability

- Important to understand the variability in your data
- Does the amount spent vary a lot among users?
  - If it does not then it will be easier to detect a change



Standard deviation

- `DataFrame.std()` : Calculate the standard deviation of a pandas DataFrame

```
# Calculate the standard deviation of revenue per user
revenue_variation = total_revenue.price.std()
print(revenue_variation)
```

```
17.520
```

Variability of revenue per user

```
# Calculate the standard deviation of revenue per user
revenue_variation = total_revenue.price.std()
```

```
17.520
```

- Good to contextualize standard deviation (sd) by calculating: mean / standard deviation?

```
revenue_variation / avg_revenue
```

```
1.084
```

Variability of purchases per user

```
# Find the average number of purchases per user
avg_purchases = total_purchases.purchase.mean()
```

```
3.15
```

```
# Find the variance in the number of purchases per user
purchase_variation = total_purchases.purchase.std()
```

```
2.68
```

```
purchase_variation / avg_purchases
```

```
0.850
```

Choosing experimental unit & response variable

- **Primary Goal:** Increase revenue
- **Better Metric:** Paywall view to purchase conversion rate
  - more granular than overall revenue
  - directly related to the our test
- **Experimental Unit:** Paywall views
  - simplest to work with
  - assuming these interactions are independent

Finding our baseline conversion rate

- **Baseline conversion rate:** Conversion rate *before* we run the test

```
# Aggregate our data sets
purchase_data = demographics_data.merge(
    paywall_views, how='inner', on=['uid']
)
# conversion rate = total purchases / total paywall views
conversion_rate = (sum(purchase_data.purchase) /
    purchase_data.purchase.count())
print(conversion_rate)
```

```
0.347
```

Let's practice!

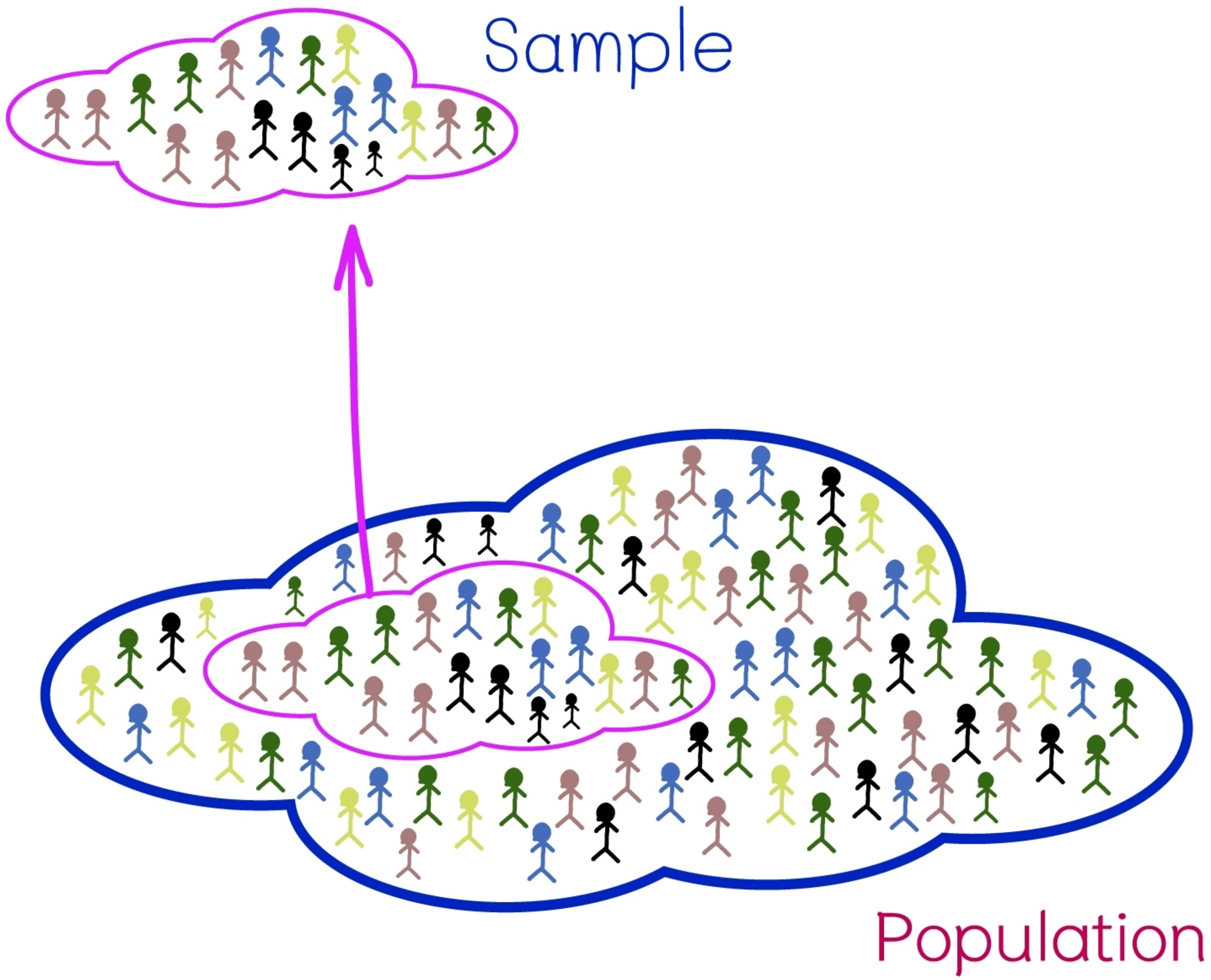
CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON

Calculating sample size

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON

Ryan Grossman
Data Scientist, EDO

Calculating the sample size of our test



Null hypothesis

- Hypothesis that control & treatment have the same impact on the response
  - Updated paywall does not improve conversion rate
  - Any observed difference is due to randomness
- Rejecting the Null Hypothesis
  - Determine their is a difference between the treatment and control
  - Statistically significant result

Types of error & confidence level

- **Confidence Level:** Probability of **not** making Type 1 Error
- Higher this value, larger test sample needed
- Common values: 0.90 & 0.95

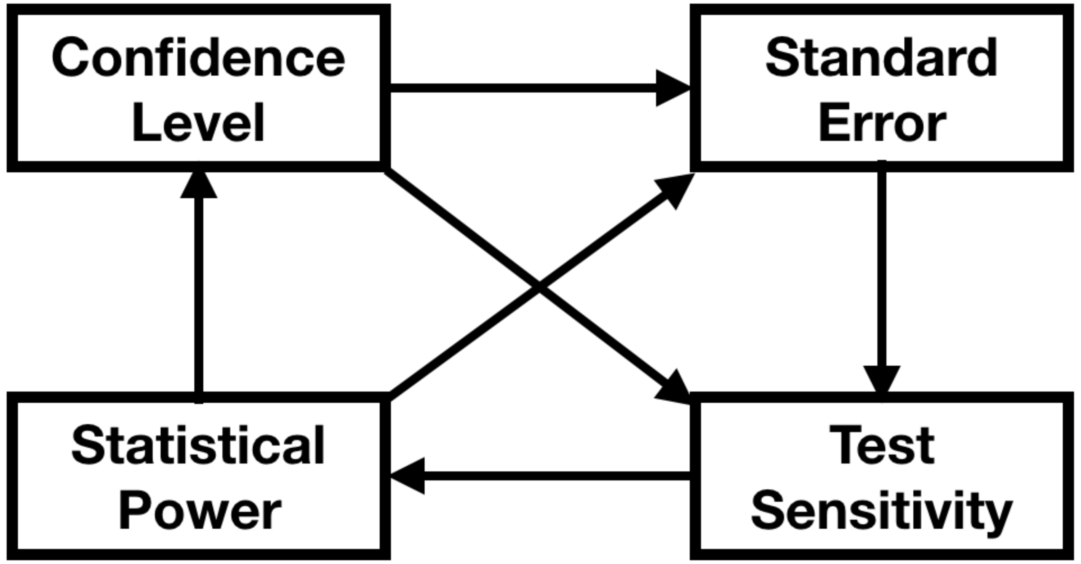
		Null Hypothesis	
		TRUE	FALSE
Null Hypothesis	Accept	Correct	Type II Error
	Reject	Type I Error	Correct

Statistical power

Statistical Power: Probability of finding a statistically significant result when the Null Hypothesis is **false**

Connecting the Different Components

- Estimate our needed sample size from:
  - needed level of sensitivity
  - our desired test power & confidence level



Power formula

- Sample size increases = Power increases
- Confidence level increases = Power decreases

$$\alpha = 1 - \textbf{confidence level}$$

$$p_1 = \textbf{Base Rate}, \; p_2 = \textbf{Base Rate} + \textbf{Sensitivity Lift}$$

$$qu = \Phi^{-1}\left(1 - \frac{\alpha}{2}\right)$$

$$diff = |p_1 - p_2|, \quad \bar{p} = \frac{(p_1 + p_2)}{2}$$

$$v_1 = p_1 \times (1 - p_1), \; v_2 = p_2 \times (1 - p_2), \quad \bar{v} = \bar{p} \times (1 - \bar{p})$$

$$Power = \Phi\left(\frac{\sqrt{n} \times diff - qu \times \sqrt{2\bar{v}}}{\sqrt{v_1 + v_2}}\right) + 1 - \Phi\left(\frac{\sqrt{n} \times diff + qu \times \sqrt{2\bar{v}}}{\sqrt{v_1 + v_2}}\right)$$

Sample size function

```
# Calculate the test power (some details omitted)
def get_power(n, p1, p2, cl):
    alpha = 1 - cl
    qu = stats.norm.ppf(1 - alpha/2)
    diff = abs(p2 - p1)
    bp = (p1 + p2) / 2
    ...
    power = power_part_one + power_part_two
    return(power)

# Calculate the sample size needed for the specified
# power and confidence level
def get_sample_size(power, p1, p2, cl, max_n = 1000000):
    n = 1
    while n <= max_n:
        tmp_power = get_power(n, p1, p2, cl)
        if tmp_power >= power:
            return n
        else:
            n = n + 1
```

Calculating our needed sample size

- **Baseline Conversion Rate:** 0.03468 (calculated previously)
- **Confidence Level:** 0.95 (chosen by us)
- **Desired Power:** 0.80 (chosen by us)
- **Sensitivity:** 0.1 (chosen by us)

```
sample_size_per_group = get_sample_size(
    0.8 # Desired Power
    conversion_rate,
    conversion_rate * 1.1 # Lifted conversion rate,
    0.95 # Confidence level)
print(sample_size_per_group)
```

```
45788
```

Generality of this function

- Function shown specific to conversion rate calculations
- Different response variables have different but analogous formulas

Decreasing the needed sample size

- Choose a unit of observation with lower variability
- Excluding users irrelevant to the process/change
- Think through how different factors relate to the sample size

Let's practice!

CUSTOMER ANALYTICS AND A/B TESTING IN PYTHON