CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

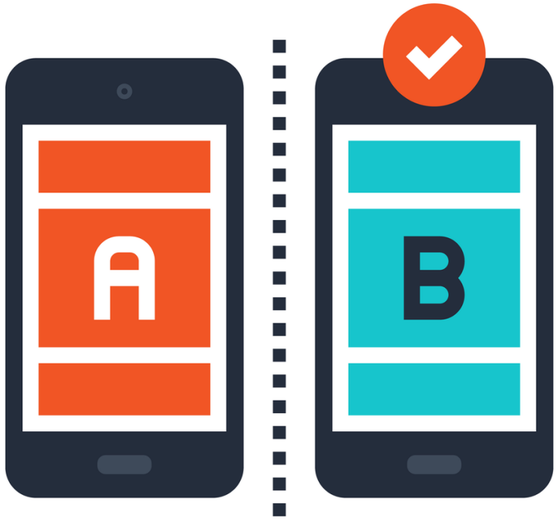
Course introduction and overview

Ryan Grossman
Data Scientist, EDO

What Is A/B Testing?

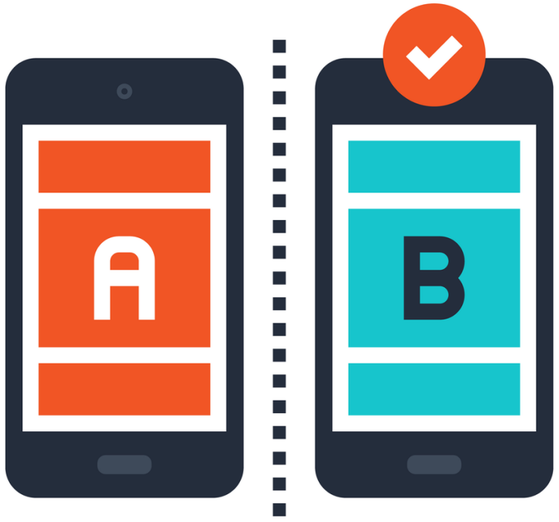
- Test two or more different ideas against each other
- See which one empirically performs better

Why is A/B Testing Important?



- No guessing

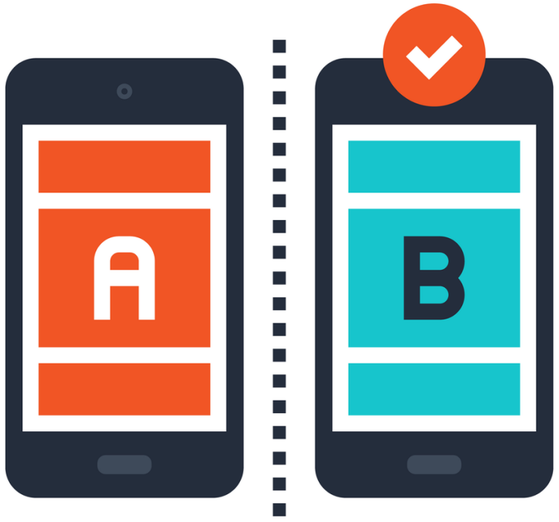
Why is A/B Testing Important?



A/B TESTING

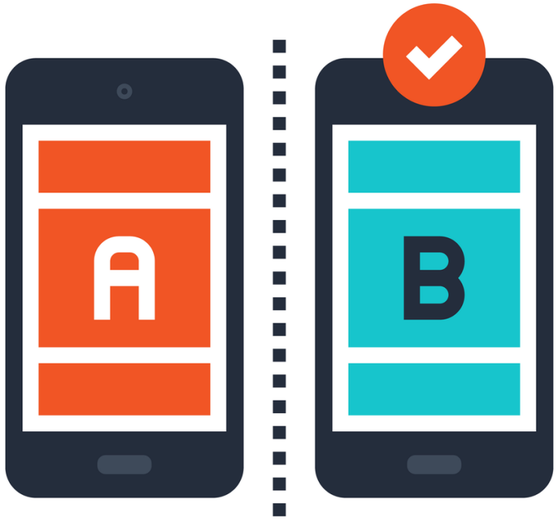
- Accurate answers

Why is A/B Testing Important?



- Rapidly iterate on ideas

Why is A/B Testing Important?



- Establish causal relationships

How Does A/B Testing Work?

Where Can A/B Testing be Used?

Course Progression

1. Understanding Users - *Key Performance Indicators*
2. Identifying Trends - *Exploratory Data Analysis*
3. Optimizing Performance - *Design of A/B Tests*
4. Data Driven Decisions - *Analyzing A/B Test Results*

Key Performance Indicators

Uncovering KPIs

Experience

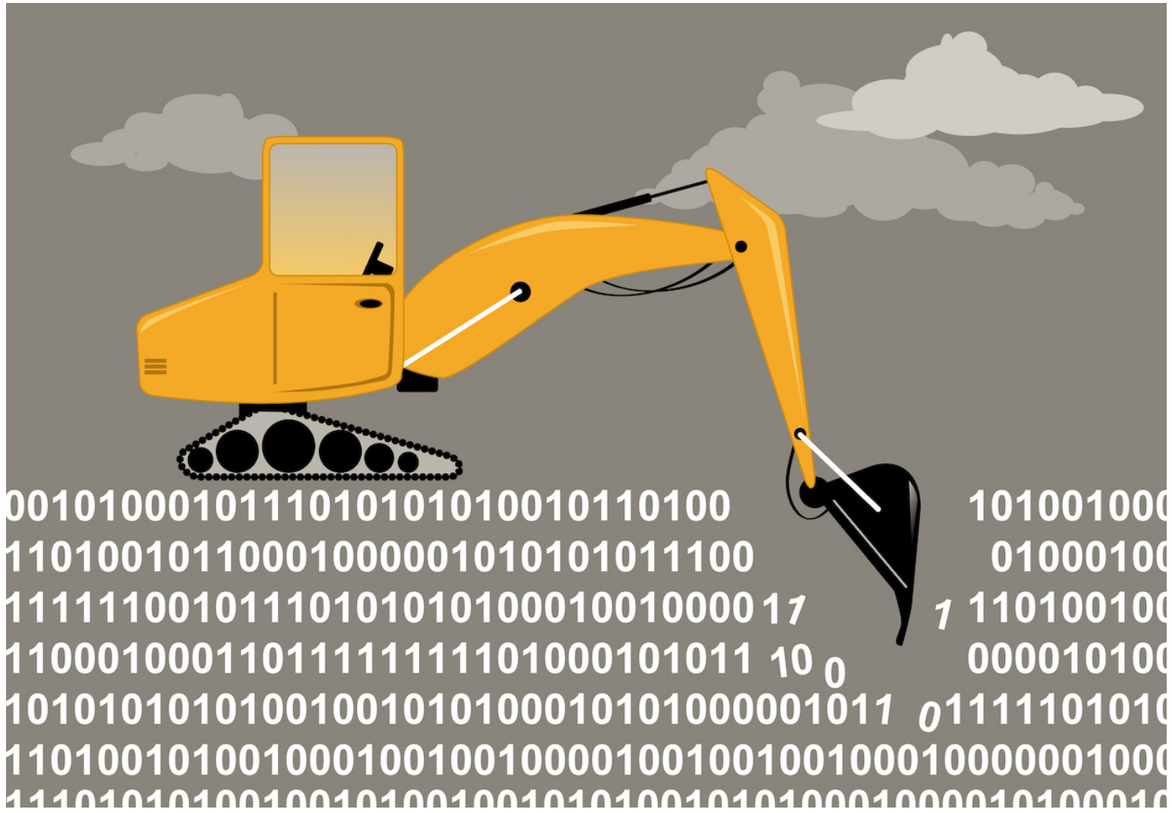
Uncovering KPIs

Domain Knowledge

Uncovering KPIs

Exploratory Data Analysis

Exploring Data



Customer dataset

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Identifying and understanding KPIs

Ryan Grossman
Data Scientist, EDO

Mobile app that offers meditation services

- Paid subscription
- One-off in-app purchases

Generalizability of A/B Testing

- The same techniques can be generalized

Dataset 1: User Demographics

```
In [1]: import pandas as pd

In [2]: customer_demographics = pd.read_csv('customer_demographics.csv')

In [3]: customer_demographics.head()

Out[3]:
uid          reg_date    device    gender    country       age
54030035     2017-06-29    and        M         USA         19
72574201     2018-03-05    iOS        F         TUR         22
64187558     2016-02-07    iOS        M         USA         16
92513925     2017-05-25    and        M         BRA         41
99231338     2017-03-26    iOS        M         FRA         59
```

Dataset 2: User Actions

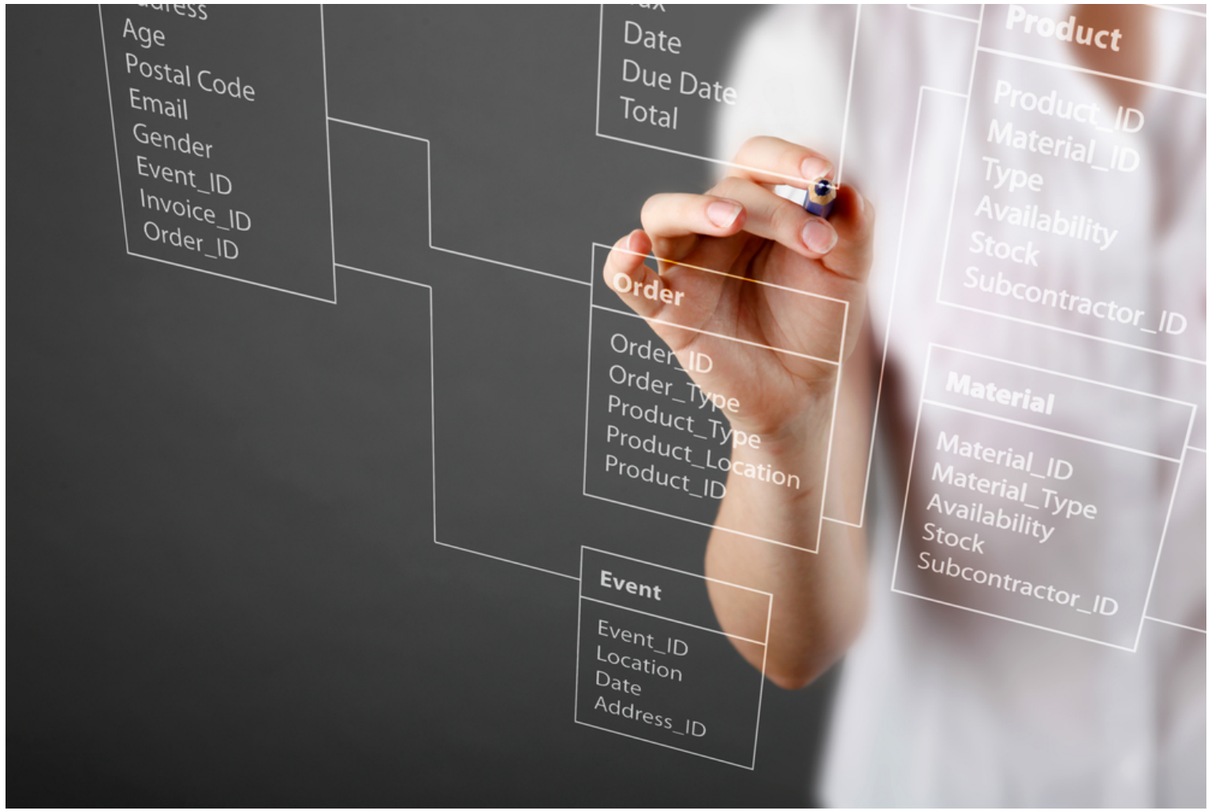
```
In [4]: customer_subscriptions = pd.read_csv('customer_subscriptions.csv')

In [5]: customer_subscriptions.head()

Out[5]:
uid         lapse_date     subscription_date      price
59435065    2017-07-06     2017-07-08             499
26485969    2018-03-12     None                   0
64187658    2016-02-14     2016-02-14             499
99231339    2017-04-02     None                   0
64229717    2017-05-24     2017-05-25             499
```

KPI: Conversion Rate

Defining Our KPI

- Stable, generalizable KPIs are better than custom KPIs

Merging the datasets



- pd.merge(df1, df2)
- df1.merge(df2)

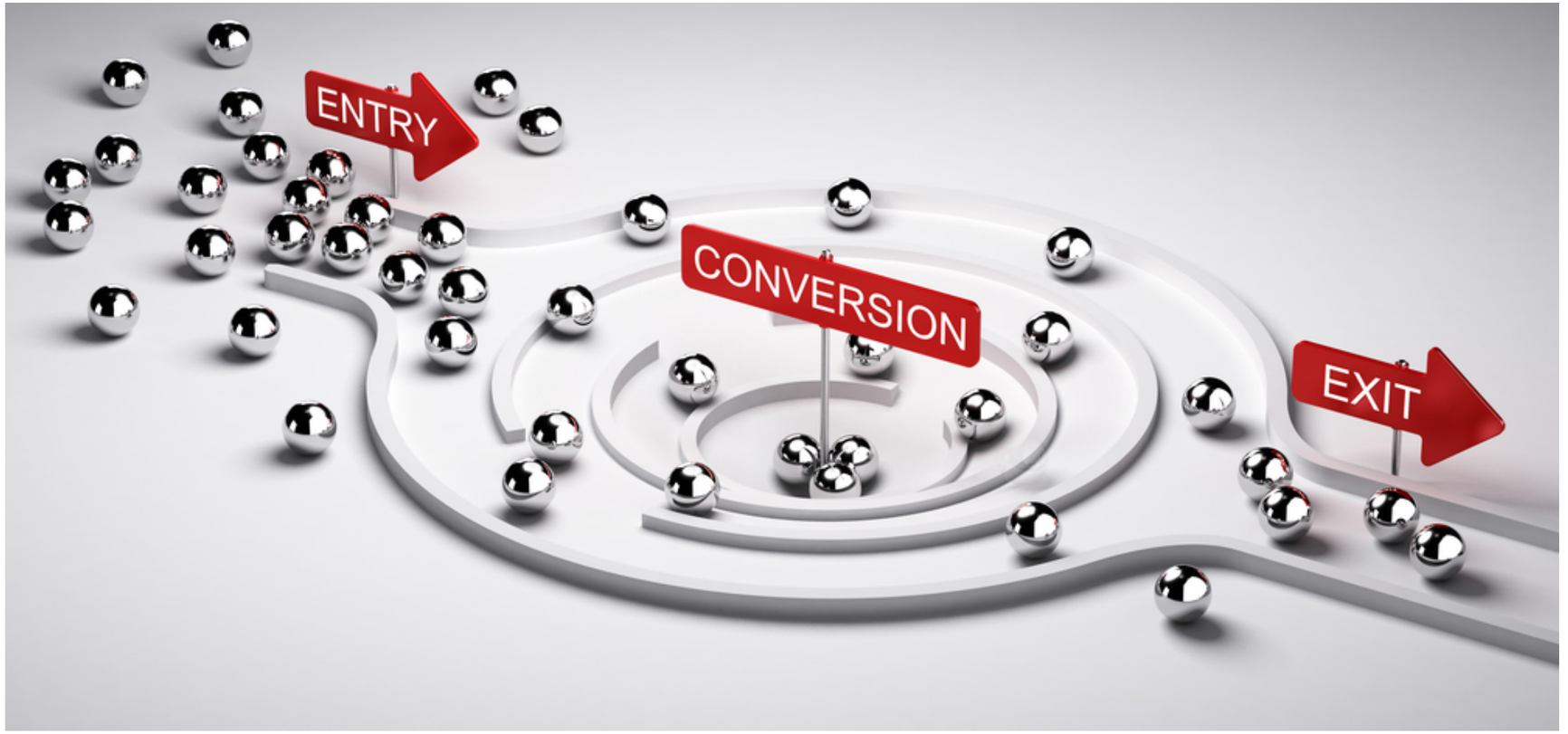
Merging Mechanics

Merge Components

```
In [6]: sub_data_demo = customer_demographics.merge(...)

In [6]: sub_data_demo = customer_demographics.merge(
                            customer_subscriptions, ...)

In [6]: sub_data_demo = customer_demographics.merge(
                            customer_subscriptions,
                            how='inner', ....)

In [6]: sub_data_demo = customer_demographics.merge(
                            customer_subscriptions,
                            how='inner', on=['uid'])
```

```
In [7]: sub_data_demo.head()

Out[7]:
uid          reg_date         device   ...price
54030729     2017-06-29       and      ...499
```

Next Steps



- Aggregate combined dataset
- Calculate the potential KPIs

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Tools for the exploratory analysis of KPIs

Ryan Grossman
Data Scientist, EDO

Reminder: Conversion rate is just one KPI

- Most companies will have many KPIs
- Each serving a different purpose
- We are working through one of these cases: Conversion rate

Methods for Calculating KPIs

`pandas.DataFrame.groupby`

```
DataFrame.groupby(by=None, axis=0, level=None,
            as_index=True, sort=True,
            group_keys=True, squeeze=False, **kwargs)
```

`pandas.DataFrame.agg`

```
DataFrame.agg(func, axis=0, *args, **kwargs)
```

pandas .groupby()

```
In [1] sub_data_grp = sub_data_demo.groupby(...)
```

`by` Argument

```
In [1] sub_data_grp = sub_data_demo.groupby(by=['country', 'device'], ...)
```

`axis` Argument

```
In [1] sub_data_grp = sub_data_demo.groupby(by=['country', 'device'],
                                            axis=0, ...)
```

`as_index` Argument

```
In [1] sub_data_grp = sub_data_demo.groupby(by=['country', 'device'],
                                            axis=0, as_index=False)
```

Aggregating Our Data

`DataFrameGroupBy` Object

```
In [2]: sub_data_grp

Out[2]: <pandas.core.groupby.DataFrameGroupBy object at 0x10ec29080>
```

Mean Price Paid

```
In [3]: sub_data_grp.price.mean()

Out[3]:
    country    device    price
0    BRA         and       312.163551
1    BRA         iOS       247.884615
2    CAN         and       431.448718
3    CAN         iOS       505.659574
4    DEU         and       398.848837
5    DEU         iOS       313.128000
6    FRA         and       320.391304
7    FRA         iOS       324.786408
```

The "Agg" Method

Simple Aggregation

```
In [4]:sub_data_grp.price.agg('mean')

Out[4]:
    country    device    price
0    BRA         and        312.163551
1    BRA         iOS        247.884615
2    CAN         and        431.448718
3    CAN         iOS        505.659574
4    DEU         and        398.848837
5    DEU         iOS        313.128000
6    FRA         and        320.391304
7    FRA         iOS        324.786408
```

The "Agg" Method

Expanded Aggregation

```
In [5]:sub_data_grp.price.agg(['mean', 'median'])

Out[5]:
                        mean      median
country device
BRA     and          312.163551      0
        iOS          247.884615      0
CAN     and          431.448718      699
        iOS          505.659574      699
DEU     and          398.848837      499
        iOS          313.128000      0
FRA     and          320.391304      0
        iOS          324.786408      0
TUR     and          216.622951      0
        iOS          249.638462      0
USA     and          420.124650      499
        iOS          380.463265      499
```

The "Agg" Method

Full Aggregation

```
In [6]: sub_data_grp.agg({'price': ['mean', 'min', 'max'],
            'age': ['mean', 'min', 'max']})

Out[6]
      country device  price                    age
                      mean          min   max  mean          min   max
0     BRA     and     312.163551    0     999  24.303738     15    67
1     BRA     iOS     247.884615    0     999  24.024476     15    79
2     CAN     and     431.448718    0     999  23.269231     15    58
3     CAN     iOS     505.659574    0     999  22.234043     15    38
4     DEU     and     398.848837    0     999  23.848837     15    67
5     DEU     iOS     313.128000    0     999  24.208000     15    54
6     FRA     and     320.391304    0     999  24.808696     15    55
7     FRA     iOS     324.786408    0     999  25.475728     15    62
8     TUR     and     216.622951    0     999  24.704918     15    59
9     TUR     iOS     249.638462    0     999  23.623077     15    65
10    USA     and     420.124650    0     999  25.000000     15    81
11    USA     iOS     380.463265    0     999  25.146939     15    84
...
```

Custom Aggregation

Truncating Function

```
In [7]: def truncated_mean(data):
    top_val = data.quantile(.9)
    bot_val = data.quantile(.1)

    trunc_data = data[(data <= top_val) & (data >= bot_val)]
    mean = trunc_data.mean()

    return(mean)
```

Custom Function Aggregation

```
In [8]: sub_data_grp.agg({'age': [truncated_mean]})

Out[8]:
    country    device    age
                            truncated_mean
0    BRA         and                22.636364
...
```

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

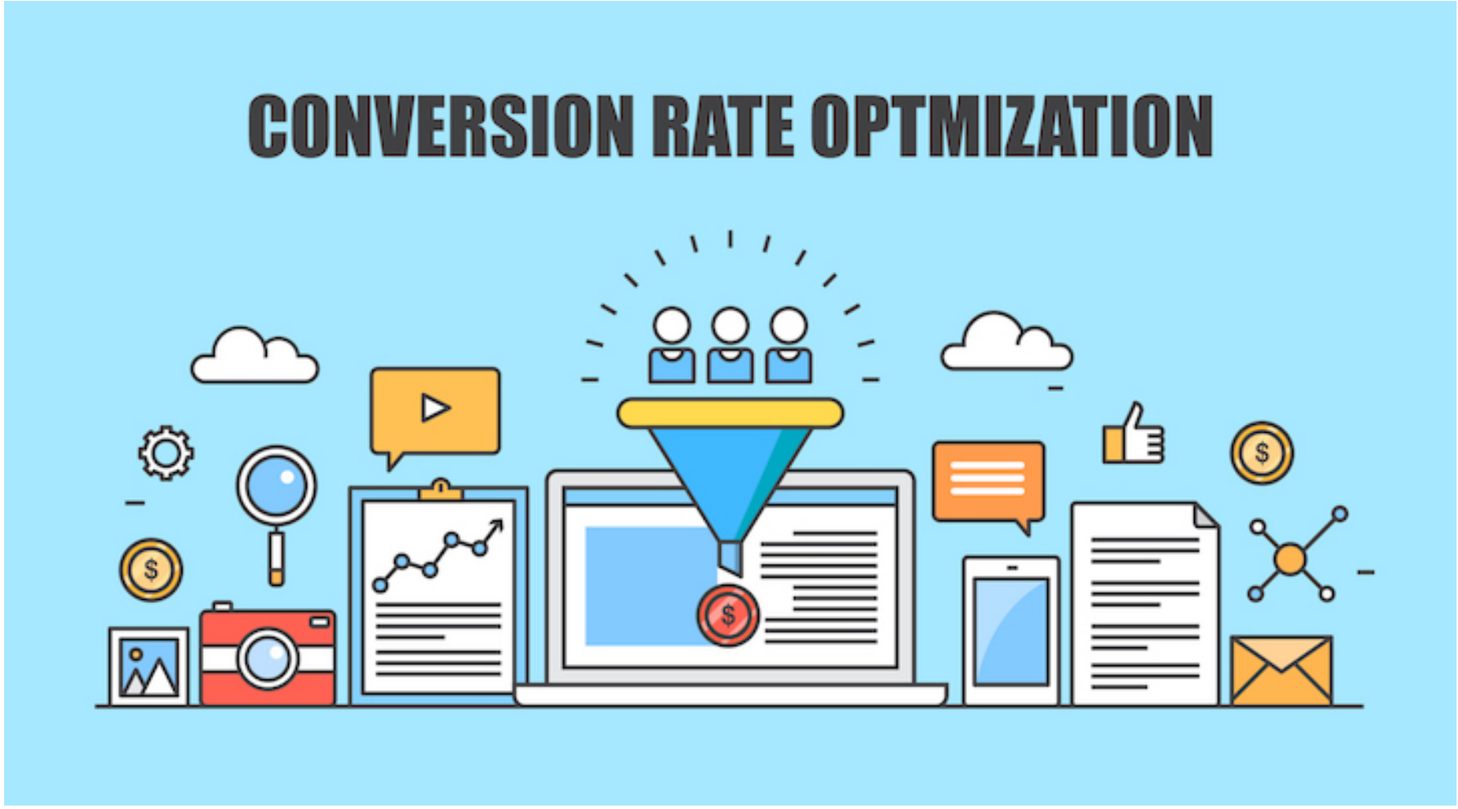
Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Calculating KPIs - A practical example

Ryan Grossman
Data Scientist, EDO

Goal - Comparing Our KPIs



- Goal: Examine KPI of user conversion rate after free trial ends
- In this case: Look at first week after the trial ends

KPI Calculation

```
In [1]: import numpy as np

In [2]: from datetime import datetime, timedelta

In [3]: sub_data_demo.lapse_date.max()
Out[3]: '2018-03-17'
```

- "Lapse date" = Date that the trial ended

KPI Calculation

- Remove users who lapsed today or any of the prior 7 days
- Ensures everyone has a full 7 days to subscribe

```
In [4]: current_date = pd.to_datetime('2018-03-17')

In [5]: max_lapse_date = current_date - timedelta(days=7)

In [6]: conv_sub_data = sub_data_demo[(sub_data_demo.lapse_date
                                       < max_lapse_date)]
```

KPI Calculation

```
In [7]: total_users = conv_sub_data.price.count()

Out[7]: 2787
```

```
In [8]: sub_days = conv_sub_data.lapse_date + timedelta(days=7)

In [9]: total_subs = conv_sub_data[
                        (conv_sub_data.price > 0) &
                        (conv_sub_data.subscription_date <= sub_days)]

In [10]: total_subs = total_subs.price.count()

In [11]: total_subs
Out[11]: 648
```

KPI Calculation

```
In [12]: conversion_rate = total_subs / total_users

In [13]: conversion_rate

Out[13]: 0.23250807319698599
```

Cohort Conversion Rate

```
In [14]: max_lapse_date = current_date - timedelta(days=14)

In [15]: conv_sub_data = conv_sub_data.copy()
In [16]: conv_sub_data = sub_data_demo[(sub_data_demo.lapse_date
                                <= max_lapse_date)]
```

Cohort Conversion Rate

```
In [17]: sub_time = np.where(
             conv_sub_data.subscription_date.notnull(),
             (conv_sub_data.subscription_date -
             conv_sub_data.lapse_date).dt.days,
             pd.NaT)

In [18]: conv_sub_data['sub_time'] = sub_time
```

Cohort Conversion Rate

```
In [20]: purchase_cohorts = conv_sub_data.groupby(
             by=['gender', 'device'], as_index=False)

In [21]: purchase_cohorts.agg({sub_time: [gcr7,gcr14]})

Out[21]:
    gender    device    sub_time
                gcr7    gcr14
0     F     and    0.221963    0.230140
1     F     iOS    0.229310    0.237931
2     M     and    0.252349    0.257718
3     M     iOS    0.218045    0.225564
```


KPI

How long does it take to gain insight into a metric?

- Monthly conversion rate
  - Would need to wait a month from the lapse date
  - Impractical

Exploratory data analysis

- Can reveal relationships between metrics and key results
- Can be tied to business metrics in important ways

Why is conversion rate important?

- Could potentially serve as a warning of potential problems later on

Next chapter: Continue exploring conversion rates

- How does it evolve over time?

Measuring KPIs across groups is crucial

- Changes can impact groups in drastically different ways

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Let's practice!

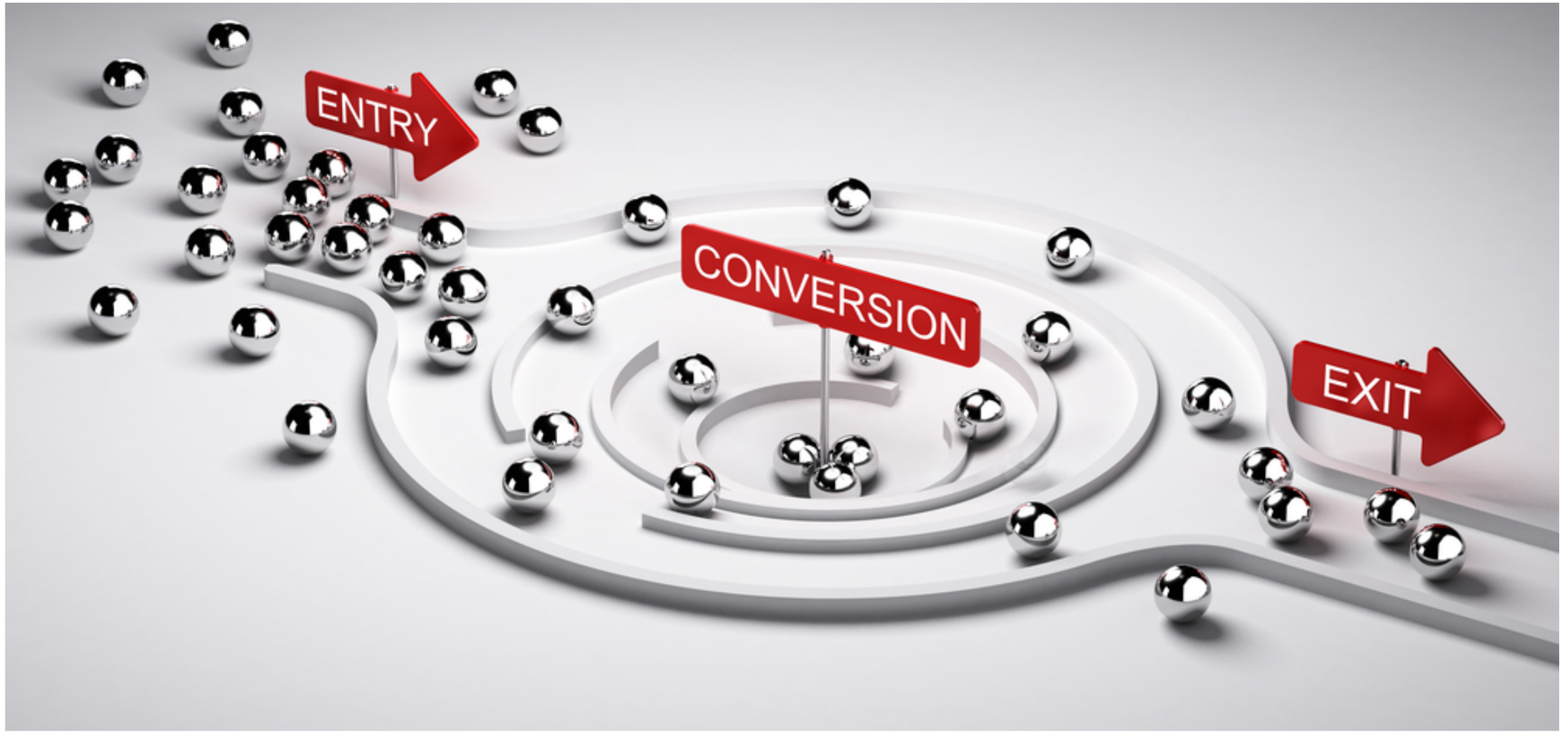
CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Working with time series data in pandas

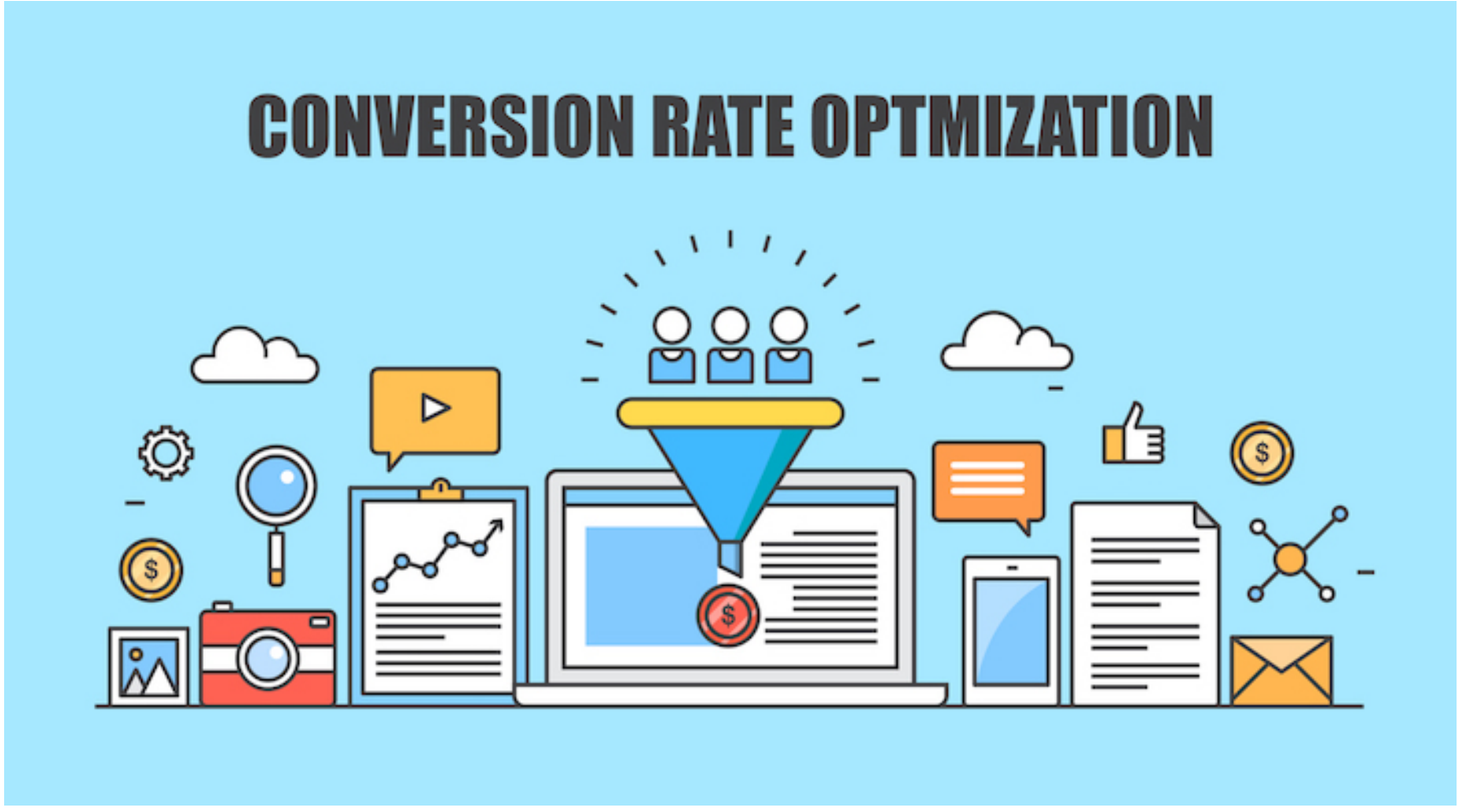
Ryan Grossman
Data Scientist, EDO

Exploratory Data Analysis

Dates & Times



Week Two Conversion Rate



Using the Timedelta Class

```
In [1]: current_date = pd.to_datetime('2018-03-17')

In [2]: max_lapse_date = current_date - timedelta(days=14)

In [3]: conv_sub_data = sub_data_demo[
    sub_data_demo.lapse_date < max_lapse_date]
```

Date Differences

```
In [4]: sub_time = (conv_sub_data.subscription_date
                    - conv_sub_data.lapse_date)

In [5]: conv_sub_data['sub_time'] = sub_time
```

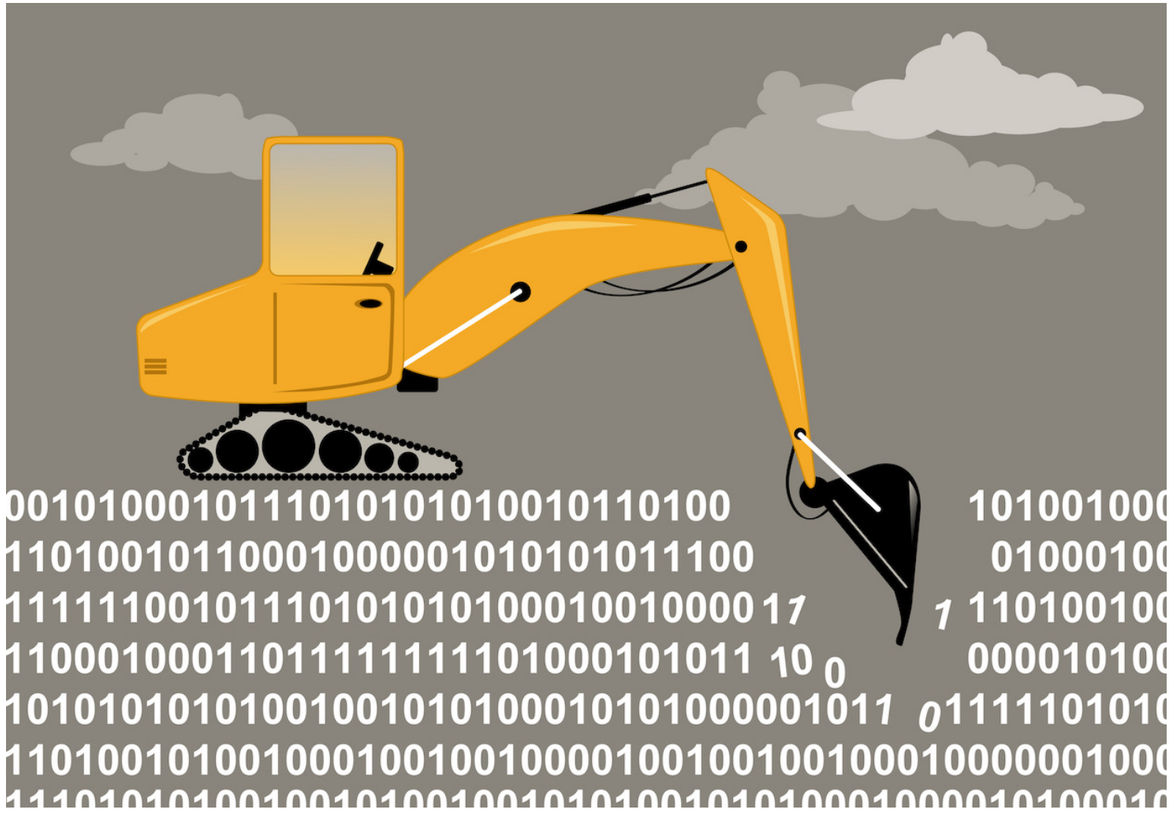
Date Components

```
In [6]: conv_sub_data['sub_time'] = conv_sub_data.sub_time.dt.days
```

Conversion Rate Calculation

```
In [7]: conv_base = conv_sub_data[
    (conv_sub_data.sub_time.notnull()) | (conv_sub_data.sub_time > 7)]

In [8]: total_users = len(conv_base)

In [9] total_users
Out[9]: 2086

In [10]: total_subs = np.where(conv_sub_data.sub_time.notnull()
    & (conv_base.sub_time <= 14), 1, 0)

In [11] total_subs = sum(total_subs)

In [12] total_subs
Out[12] 20

In [13] conversion_rate = total_subs / total_users

In [14] conversion_rate
Out[14]: 0.0095877277085330784
```

Digging Deeper



Parsing Dates - On Import

```
pandas.read_csv(...,
    parse_dates=False,
    infer_datetime_format=False,
    keep_date_col=False,
    date_parser=None,
    dayfirst=False,
    ...)
```

```
In [15]: customer_demographics = pd.read_csv('customer_demographics.csv',
    parse_dates=True,
    infer_datetime_format=True
)

Out [15]:
    uid            reg_date          device gender    country    age
0    54030035.0    2017-06-29        and        M        USA        19
1    72574201.0    2018-03-05        iOS        F        TUR        22
2    64187558.0    2016-02-07        iOS        M        USA        16
3    92513925.0    2017-05-25        and        M        BRA        41
4    99231338.0    2017-03-26        iOS        M        FRA        59
```

Parsing Dates - Manually

```
pandas.to_datetime(arg, errors='raise', ..., format=None, ...)
```

strftime

1993-01-27 --`"%Y-%m-%d"`

05/13/2017 05:45:37 -- `"%m/%d/%Y %H:%M:%S"`

September 01, 2017 -- `"%B %d, %Y"`

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Creating time series graphs With matplotlib

Ryan Grossman
Data Scientist, EDO

Conversion Rate Over Time

Monitoring The Impact of Changes

Conversion Rate by Day

```
In [1]: current_date = pd.to_datetime('2018-03-17')

In [2]: max_lapse_date = current_date - timedelta(days=7)

In [3]: conv_sub_data = sub_data_demo[sub_data_demo.lapse_date
                                < max_lapse_date]

In [4]: sub_time = (conv_sub_data.subscription_date -
                            conv_sub_data.lapse_date).dt.days

In [5]: conv_sub_data['sub_time'] = sub_time
```

Conversion Rate by Day

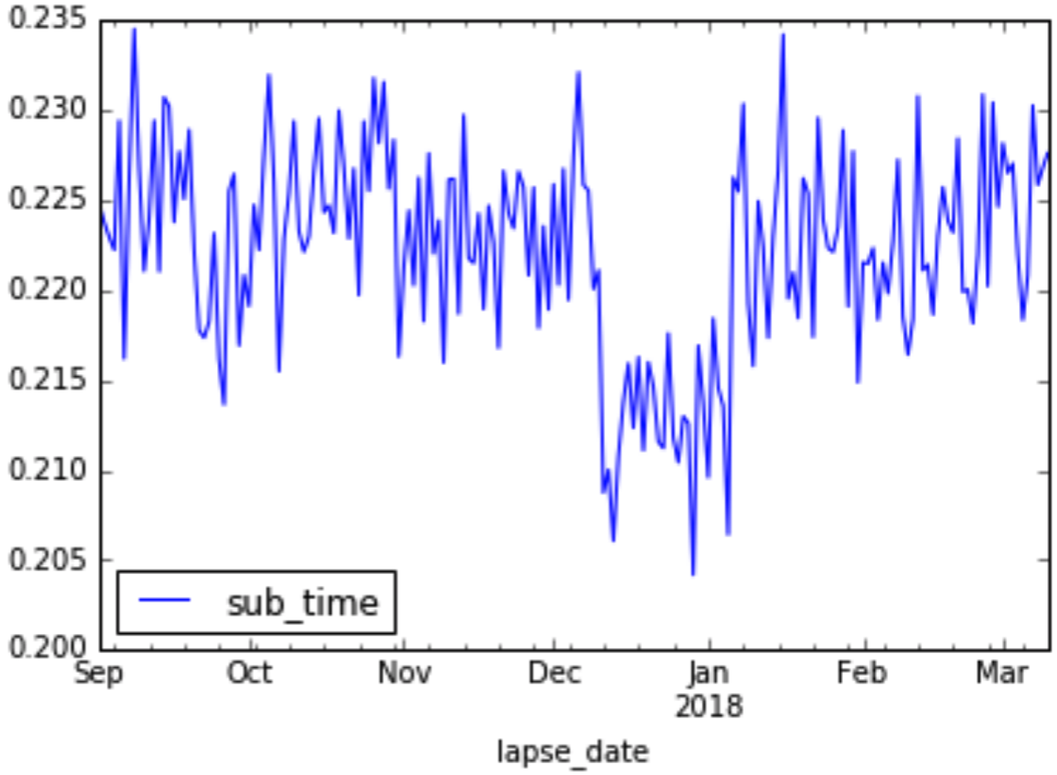
```
In [6]: conversion_data = conv_sub_data.groupby(by=['lapse_date'],
                                 as_index=False)

In [7]: conversion_data = conversion_data.agg({'sub_time': [gc7]})

In [9]: conversion_data.columns = conversion_data.columns.droplevel(
                                       level=1)

In [8]: conversion_data.head()

Out[8]:
    lapse_date    sub_time
0    2017-09-01    0.224775
1    2017-09-02    0.223749
2    2017-09-03    0.222948
3    2017-09-04    0.222222
4    2017-09-05    0.229401
```

Plotting Daily Conversion Rate

```
In [9]: conversion_data.lapse_date =
            pd.to_datetime(conversion_data.lapse_date)

In [10]: conversion_data.plot(x='lapse_date', y='sub_time')
```

Plotting Daily Conversion Rate

```
In [13]: plt.show()
```



Trends in Different Cohorts

```
In [11]: conversion_data.head()

Out[11]:
    lapse_date    country    sub_time
0    2017-09-01     BRA        0.184000
1    2017-09-01     CAN        0.285714
2    2017-09-01     DEU        0.276119
3    2017-09-01     FRA        0.240506
4    2017-09-01     TUR        0.161905
```

.pivot_table()

Pivot Table Method

```
pandas.pivot_table(
    data, values=None, index=None, columns=None,
    aggfunc='mean', fill_value=None, margins=False,
    dropna=True, margins_name='All')
```

.pivot_table()

```
In [12]: reformatted_cntry_data =pd.pivot_table(conversion_data, ...)

In [13]: reformatted_cntry_data =pd.pivot_table(conversion_data,
    values=['sub_time'],...)

In [14]: reformatted_cntry_data =pd.pivot_table(conversion_data,
    values=['sub_time'], columns=['country'],
    ...)

In [15]: reformatted_cntry_data =pd.pivot_table(conversion_data,
        values=['sub_time'],columns=['country'],
        index=['reg_date'],fill_value=0 )
```

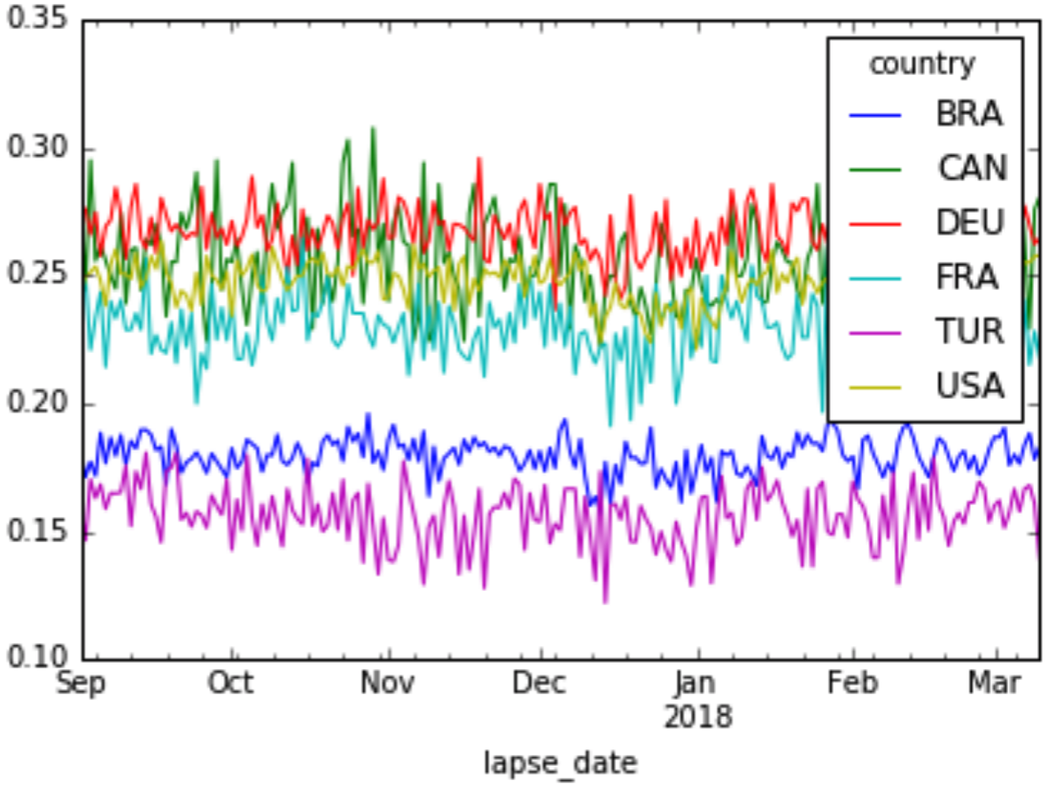
.pivot_table()

```
In [16]: reformatted_cntry_data.columns
    reformatted_cntry_data.columns.droplevel(level=[0])

In [17]: reformatted_cntry_data.reset_index(inplace=True)

In [18]: reformatted_cntry_data.head()

Out[18]:
lapse_date          BRA          CAN          DEU
2017-09-01     0.184000     0.285714     0.276119    ...
2017-09-02     0.171296     0.244444     0.276190    ...
2017-09-03     0.177305     0.295082     0.266055    ...
```

Plotting Trends in Different Cohorts

```
In [19]: reformatted_cntry_data.plot(
     x='reg_date',
     y=['BRA','FRA','DEU','TUR','USA','CAN']
)

In [20]: plt.show()
```

Plotting Trends in Different Cohorts



CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Understanding and visualizing trends in customer data

Ryan Grossman
Data Scientist, EDO

Further Techniques for Uncovering Trends

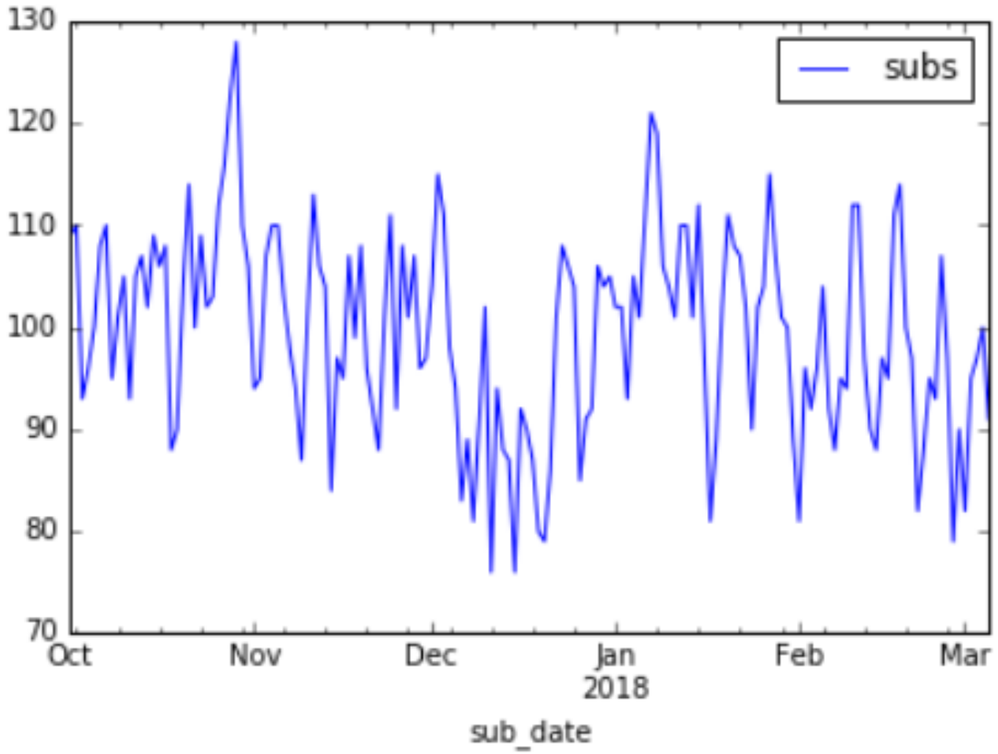
Subscribers Per Day

```
In [1]: usa_subscriptions = pd.read_csv('usa_subscribers.csv',
                                        parse_dates=True,
                                        infer_datetime_format=True)

In [2]: usa_subscriptions['sub_day'] = (usa_subscriptions.sub_date -
                                        usa_subscriptions.lapse_date).dt.days

In [3]: usa_subscriptions = usa_subscriptions[
                                        usa_subscriptions.sub_day <= 7]

In [4]: usa_subscriptions = usa_subscriptions.groupby(
                                        by=['sub_date'], as_index = False)

In [5]: usa_subscriptions = usa_subscriptions.agg({'subs': ['sum']})

In [6]: usa_subscriptions.columns = usa_subscriptions.columns.droplevel(
          level=[1])

In [7]: usa_subscriptions.head()

Out[7]:
    sub_date         subs
0    2016-09-02        37
1    2016-09-03        50
2    2016-09-04        59
```

Subscribers Per Day

```
In [8]: usa_subscriptions.plot(x='sub_date', y='subs')

In [9]: plt.show()
```

Seasonality



Correcting for Seasonality

Trailing Averages

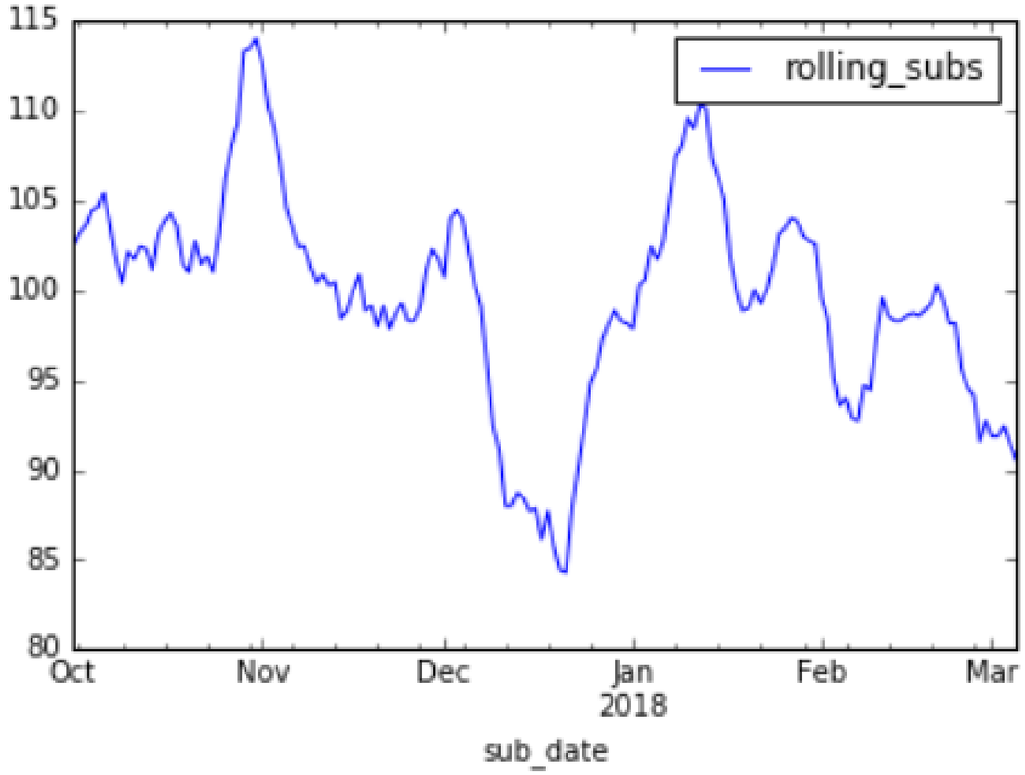
Calculating Trailing Averages

```
In [10]: rolling_subs =  usa_subscriptions.subs.rolling(...)

In [10]: rolling_subs =  usa_subscriptions.subs.rolling(window=7, ...)

In [10]: rolling_subs = usa_subscriptions.subs.rolling(
           window=7, center=False)
```

Calculating Trailing Averages

```
In [11]: rolling_subs = rolling_subs.mean()

In [12]: usa_subscriptions['rolling_subs'] = rolling_subs

In [13]: usa_subscriptions.tail()

Out[13]:
sub_date    subs        rolling_subs
2018-03-14  89          94.714286
2018-03-15  96          95.428571
2018-03-16  102         96.142857
2018-03-17  102         96.142857
2018-03-18  115         98.714286
```

Smoothed Data

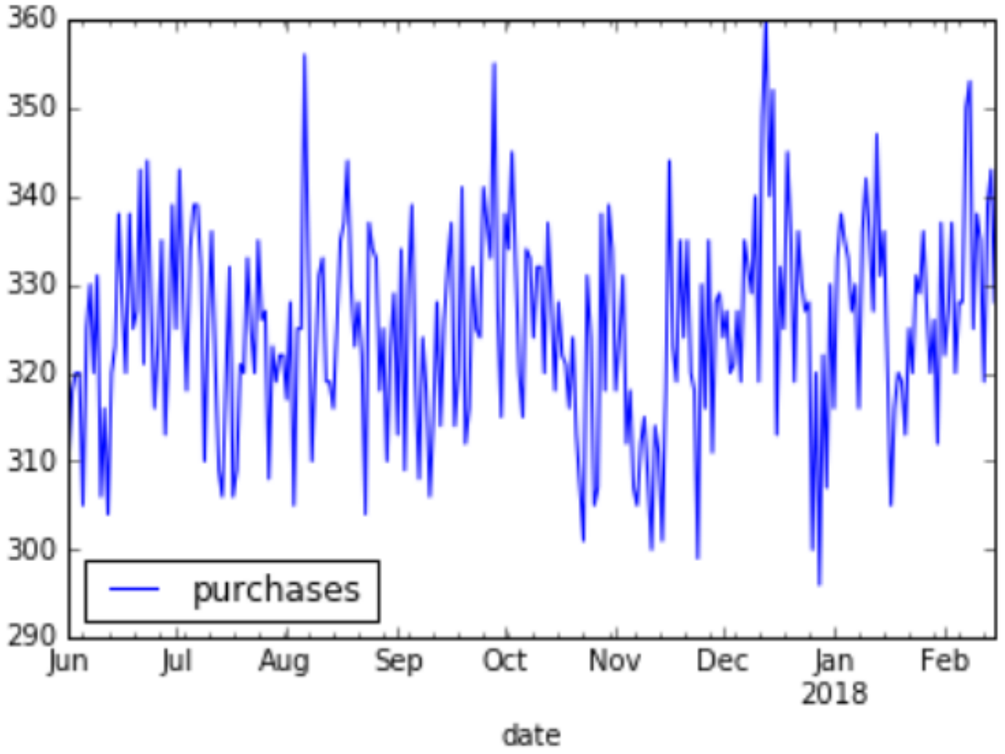


Noisy Data

Noisy Data

```
In [14]: high_sku_purchases = pd.read_csv('high_sku_purchases.csv',
                                          parse_dates=True,
                                          infer_datetime_format=True)

In [15]: high_sku_purchases.plot(x='date', y='purchases')

In [16]: plt.show()
```

Exponential Moving Average

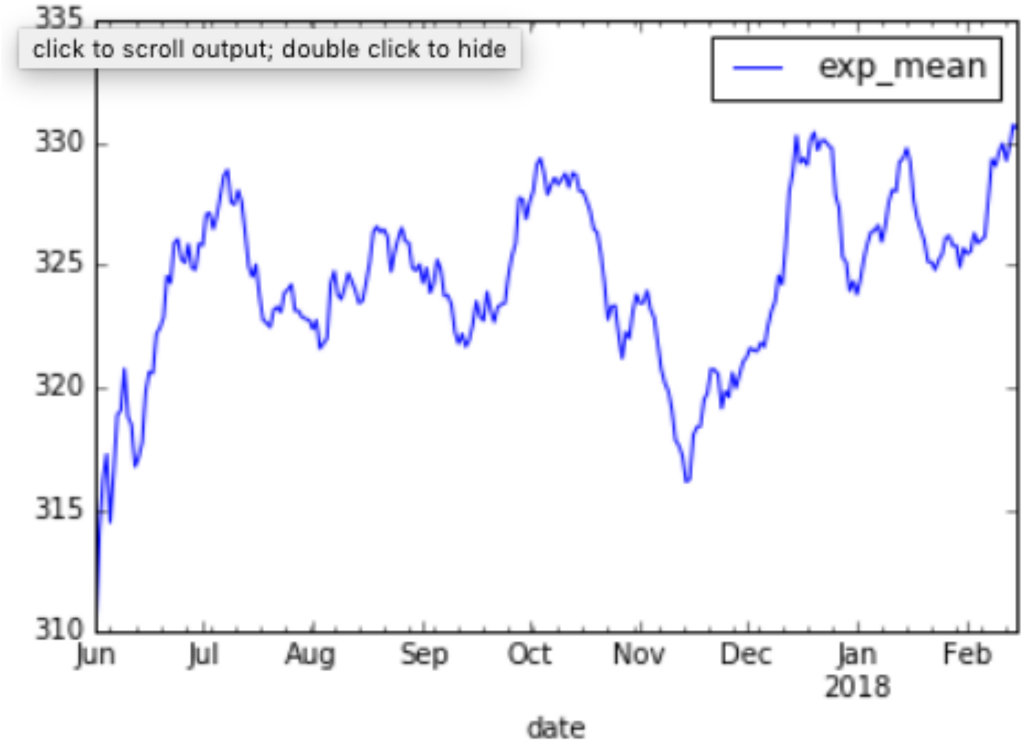


Calculating an Exponential Moving Average

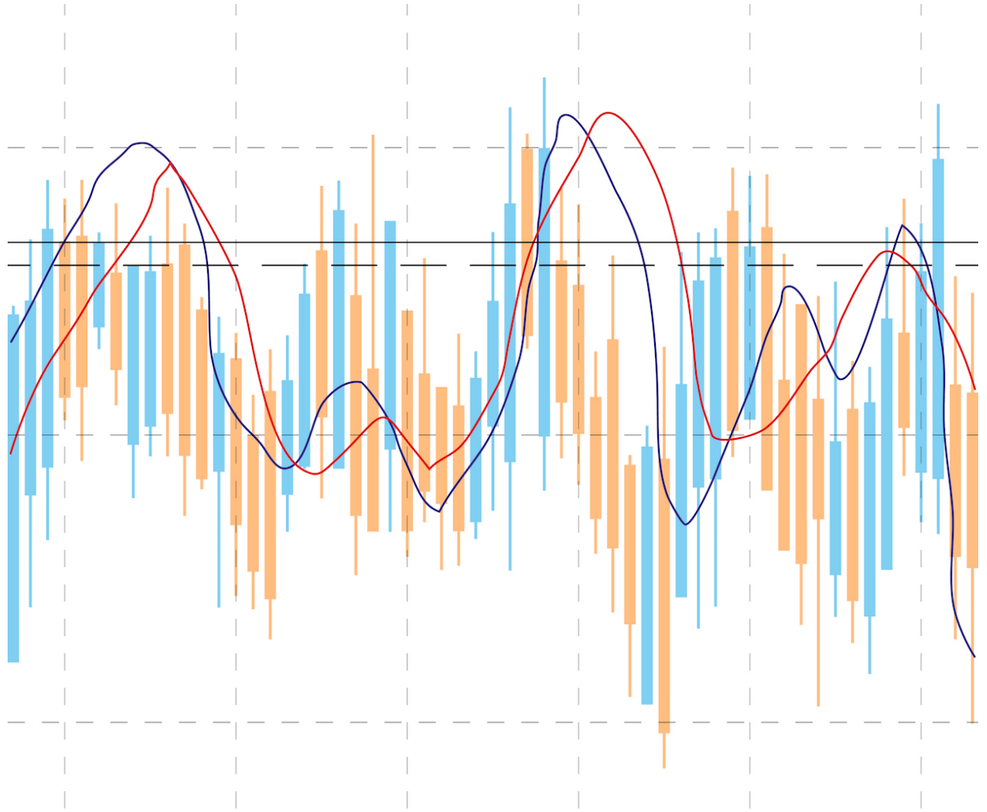
```
In [17]: exp_mean = high_sku_purchases.purchases.ewm(span=30)

In [18]: exp_mean = exp_mean.mean()

In [19]: high_sku_purchases['exp_mean'] = exp_mean
```

Calculating an Exponential Moving Average

```
In [20]: high_sku_purchases.plot(x='date', y='exp_mean')
```



Data Smoothing Techniques



CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Exploratory data analysis with time series data

Ryan Grossman
Data Scientist, EDO

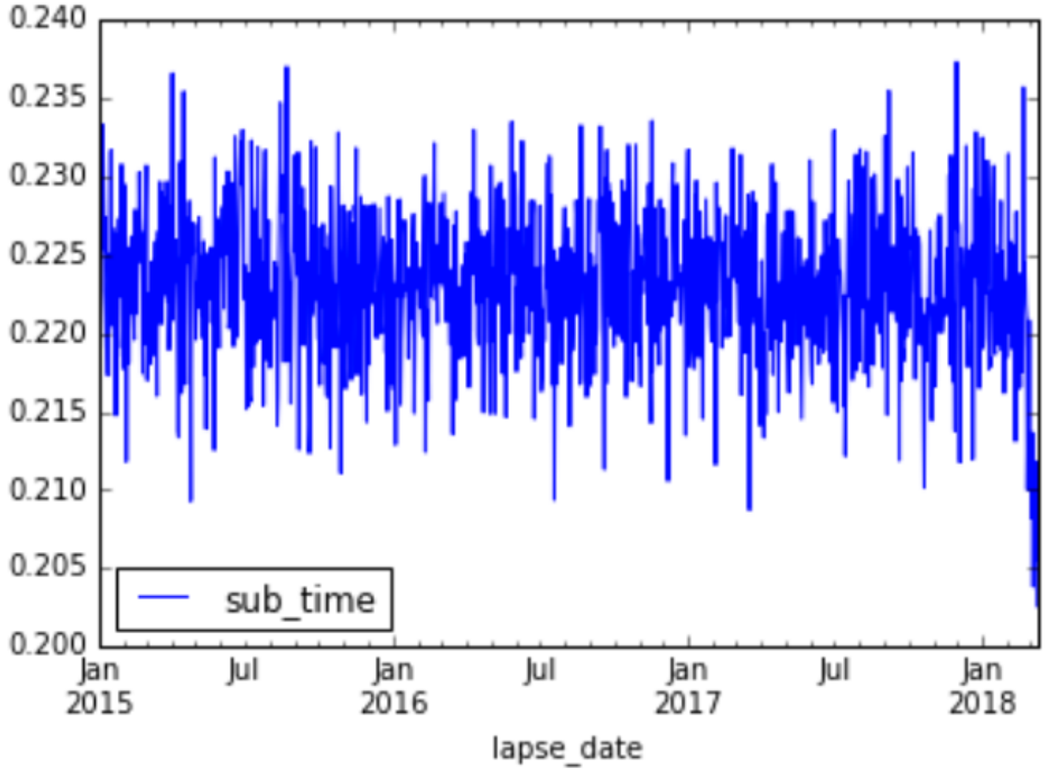
Exploratory Analysis

Drop in New User Retention

```
In [1]: current_date = pd.to_datetime('2018-03-17')
In [2]: max_lapse_date = current_date - timedelta(days=7)
In [3]: conv_sub_data = sub_data_demo[sub_data_demo.lapse_date
                                      <= max_lapse_date]

In [4]: sub_time = (conv_sub_data.subscription_date -
                    conv_sub_data.lapse_date).dt.days
In [6]: conv_sub_data['sub_time'] = sub_time

In [7]: conversion_data = conv_sub_data.groupby(by=['lapse_date'],
                                  as_index=False)
In [8]: conversion_data = conversion_data.agg({'sub_time': [gc7]})
In [9]: conversion_data.columns = conversion_data.columns.droplevel(level=1)
```

```
In [10]: conversion_data.plot()
In [11]: plt.show()
```

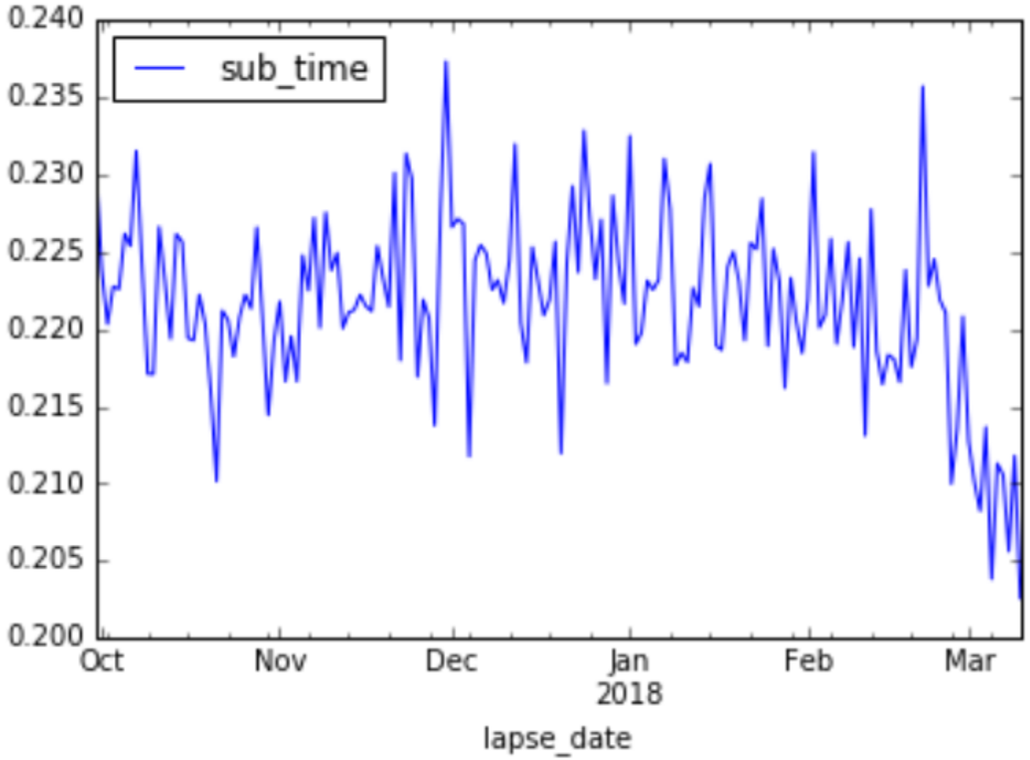
Limiting our View



Limiting our View

```
In [12]: current_date = pd.to_datetime('2018-03-17')

In [13]: start_date = current_date - timedelta(days=(6*28))

In [14]: conv_filter = ((conversion_data.lapse_date >= start_date) &
                        (conversion_data.lapse_date <= current_date))

In [15]: conversion_data_filt = conversion_data[conv_filter]

In [16]: conversion_data_filt.plot(x='lapse_date', y='sub_time')

In [17]: plt.show()
```

Uncovering the Dip



Segmenting our Graph

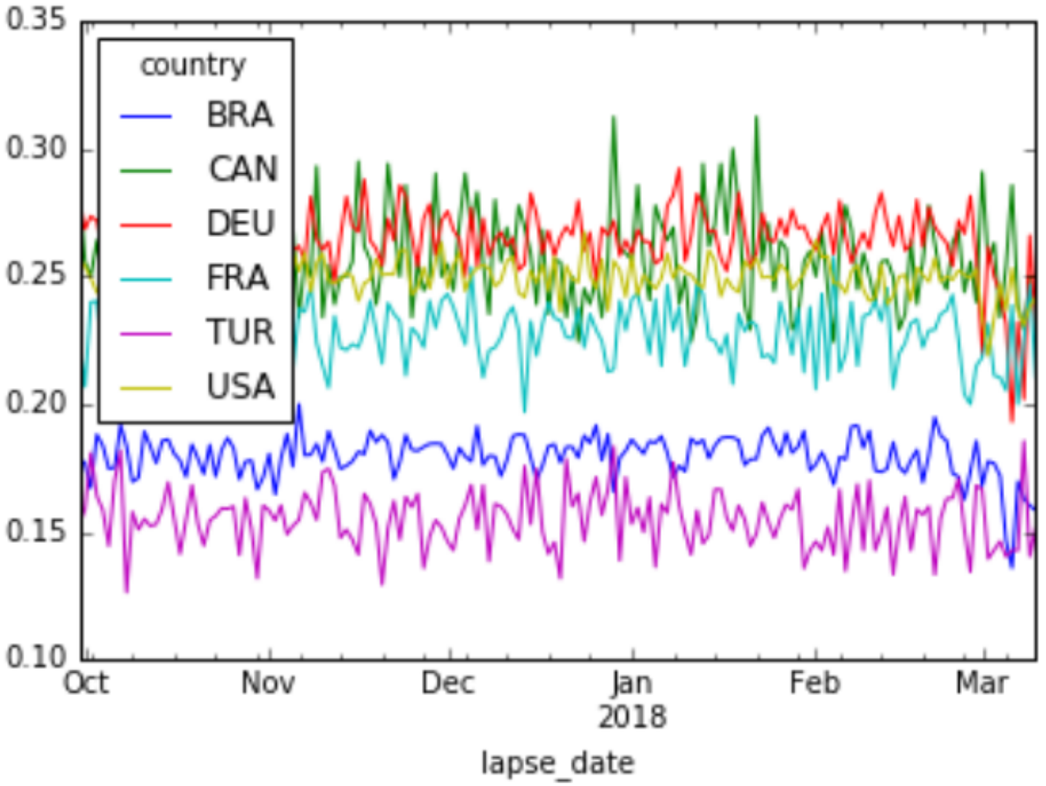
Splitting by Country & Device

```
In [18]: conv_filter = ((conv_sub_data.lapse_date >= start_date) &
                        (conv_sub_data.lapse_date <= current_date))

In [19]: conv_data = conv_sub_data[conv_filter]

In [20]: conv_data_cntry = conv_data.groupby(by=['lapse_date', 'country'],
                                as_index=False)

In [20]: conv_data_cntry = conv_data_cntry.agg({'sub_time': [gc7]})

In [21]: conv_data_cntry.columns = conv_data_cntry.columns.droplevel(level=1)

In [22]: conv_data_cntry = pd.pivot_table(conv_data_cntry,
                                values=['sub_time'],
                                columns=['country'],
                                index=['lapse_date'],fill_value=0 )

In [23]: conv_data_cntry.columns = conv_data_cntry.columns.droplevel(
                                    level=0)

In [24]: conv_data_cntry.reset_index(inplace=True)

In [25]: conv_data_cntry.plot(x=['lapse_date'],
            y=['BRA', 'CAN', 'DEU', 'FRA', 'TUR', 'USA'])

In [26]: plt.show()
```

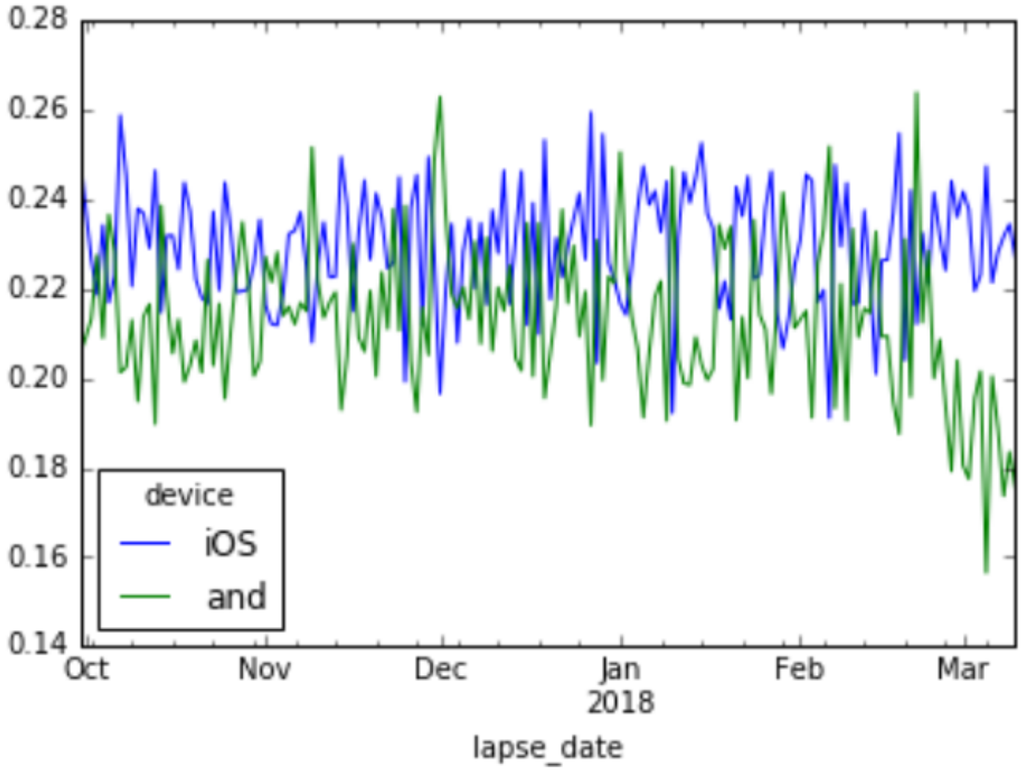
Splitting by Country & Device

```
In [27]: conv_filter = ((conv_sub_data.lapse_date >= start_date) &
                        (conv_sub_data.lapse_date <= current_date))

In [28]: conv_data = conv_sub_data[conv_filter]

In [29]: conv_data_dev = conv_data.groupby(by=['lapse_date',
                        'device'], as_index=False)

In [30]: conv_data_dev = conv_data_dev.agg({'sub_time': [gc7]})

In [31]: conv_data_dev.columns = conv_data_dev.columns.droplevel(level=1)

In [32]: conv_data_dev = pd.pivot_table(conv_data_dev, values=['sub_time'],
                        columns=['device'],
                        index=['lapse_date'],fill_value=0 )

In [33]: conv_data_dev.columns = conv_data_dev.columns.droplevel(level=[0])

In [34]: conv_data_dev.reset_index(inplace=True)

In [35]: conv_data_dev.plot(x=['lapse_date'], y=['iOS', 'and'])

In [36]: plt.show()
```

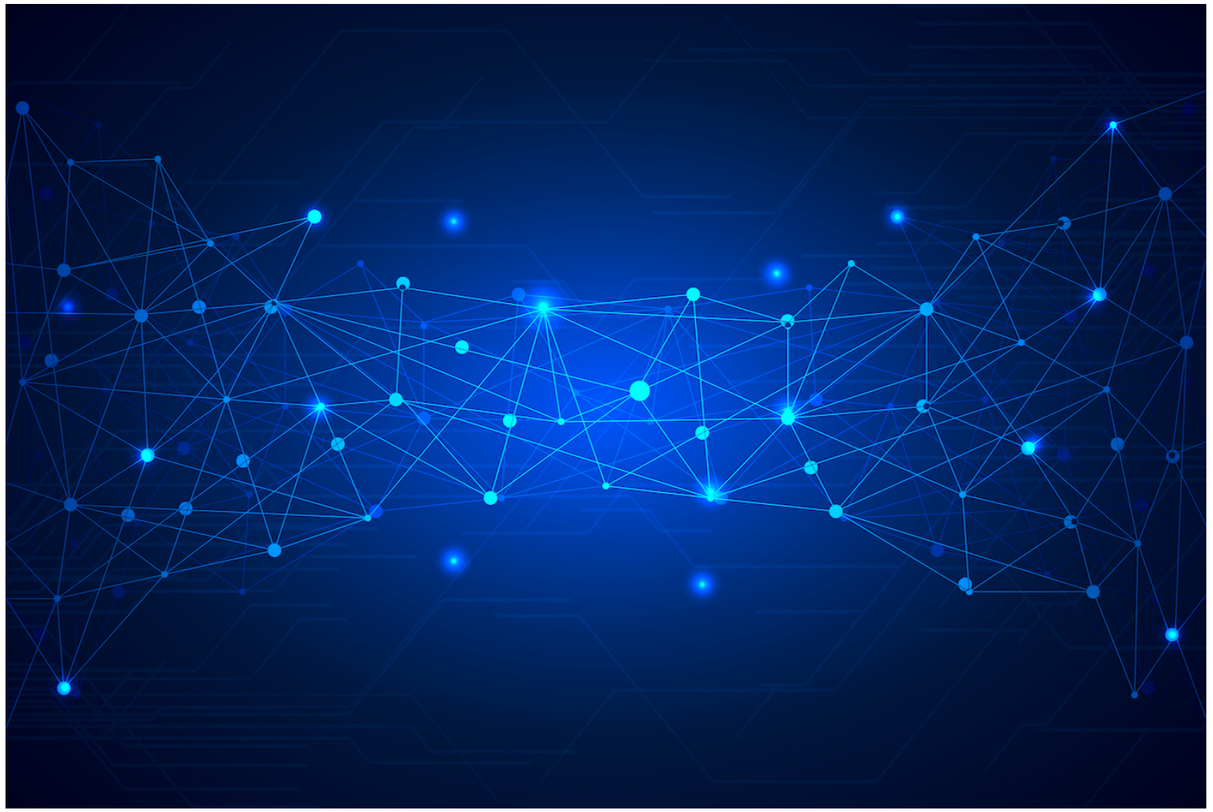
Breaking out by Country



Breaking Out by Device



Adding Annotations



Annotation Datasets

```
In [37]: events = pd.read_csv('events.csv')

In [38]: events.head()

Out[38]:
Date             Event
2018-01-01         NYD
2017-01-01         NYD
2016-01-01         NYD
2015-01-01         NYD
2014-01-01         NYD
```

```
In [39]: releases = pd.read_csv('releases.csv')

In [40]: releases.head()

Out[40]:
    Date            Event
2018-03-14       iOS Release
2018-03-03       Android Release
2018-01-13       iOS Release
2018-01-15       Android Release
2017-11-03       Android Release
```

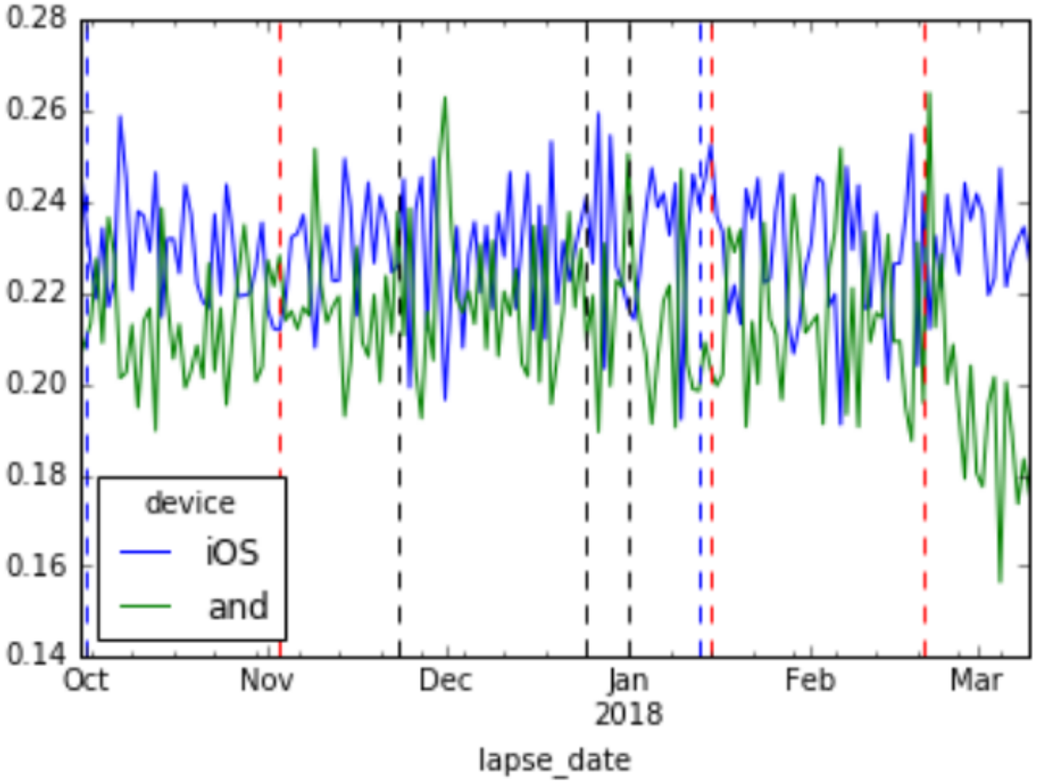
Plotting Annotations

```
In [41]: conv_data_dev.plot(x=['lapse_date'], y=['iOS', 'and'])

In [42]: events.Date = pd.to_datetime(events.Date)

In [43]: for row in events.iterrows():
    tmp = row[1]
    plt.axvline(x=tmp.Date, color='k', linestyle='--')
```

Plotting Annotations

```
In [44]: releases.Date = pd.to_datetime(releases.Date)

In [45]: for row in releases.iterrows():
    tmp = row[1]
    if tmp.Event == 'iOS Release':
        plt.axvline(x=tmp.Date, color='b', linestyle='--')
    else:
        plt.axvline(x=tmp.Date, color='r', linestyle='--')

In [46]: plt.show()
```

Our Final Plot



Exploratory Analysis

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Introduction to A/B testing

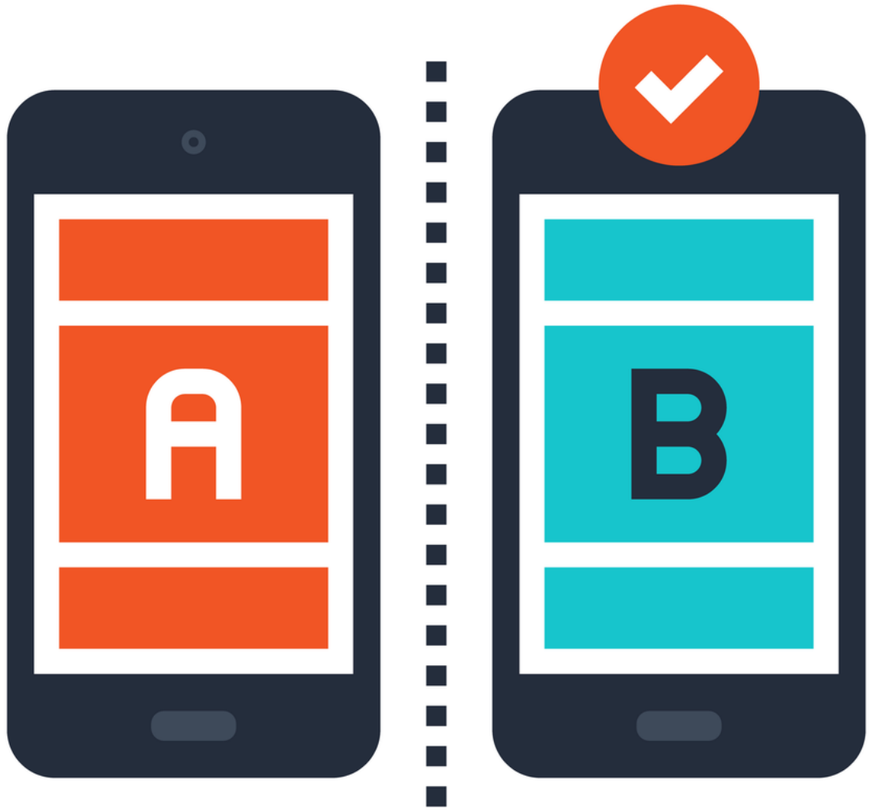
Ryan Grossman
Data Scientist, EDO

Overview


A
B

MANAGEMENT

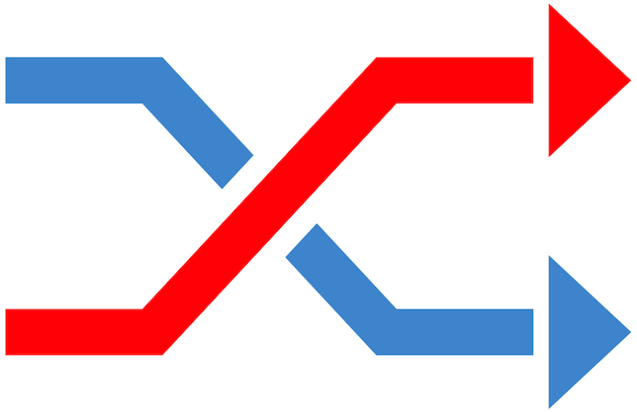
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat. Lorem ipsum dolor sit amet, consectetuer adipiscing elit.



A/B TESTING

A/B Testing Example

The Importance of Randomness



A/B Testing Flexibility

Good and Bad Cases for A/B Testing

Good and Bad Cases for A/B Testing

Good and Bad Cases for A/B Testing

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Initial A/B test design

Ryan Grossman
Data Scientist, EDO

Increasing Revenue through A/B Testing

Generalizabiliy of A/B Testing

Paywall Views & Demographics Data

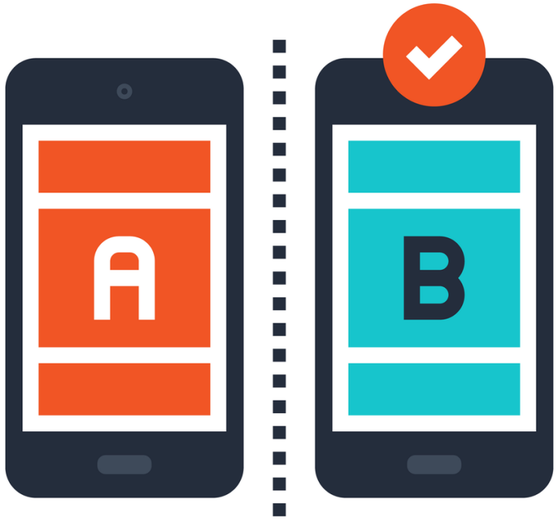
```
In [1]: demographics_data = pd.read_csv('user_demographics.csv')

In [2]: demographics_data.head(n=4)

Out[2]:
uid                reg_date    device    gender    country    age
0    52774929    2018-03-07    and        F        FRA        27
1    84341593    2017-09-22    iOS        F        TUR        22
2    41201055    2017-11-24    and        F        USA        20
3    68477880    2016-12-08    and        F        BRA        18
```

```
In [3]: paywall_views = pd.read_csv('paywall_views.csv')

In [4]: paywall_views.head(n=4)

Out [4]:
        uid        date                    purchase        sku        price
0    52774929    2018-03-11 04:11:01        0            NaN        NaN
1    52774929    2018-03-13 21:28:54        0            NaN        NaN
2    52774929    2018-03-14 12:22:17        0            NaN        NaN
3    84341593    2017-09-25 06:13:14        0            NaN        NaN
```

A/B Testing Terminology

Response Variable

Factors & Variants



Experimental Unit

Calculating Experimental Units

Calculating Experimental Units

```
In [5]: purchase_data = demographics_data.merge(paywall_views,
                                               how='left', on=['uid'])

In [6]: purchase_data_agg = purchase_data.groupby(by=['uid'],
                                                  as_index=False)

In [7]: total_purchases = purchase_data_agg.purchase.sum()

In [8]: total_purchases.purchase = np.where(
                                        np.isnan(total_purchases.purchase),
                                        0, total_purchases.purchase)

In [9]: total_purchases.purchase.mean()
```

Calculating Experimental Units

```
In [9]: total_purchases.purchase.mean()

Out[9]: 3.15
```

Calculating Experimental Units

```
In [10]: min(total_purchases.purchase)

Out[10]: 0.0

In [11]: max(total_purchases.purchase)

Out [12]: 17.0
```

Calculating Experimental Units

Calculating Experimental Units

```
In [13]: purchase_data_agg.date = paywall_views.date.dt.floor('d')

In [14]: purchase_data_agg = purchase_data.groupby(
                                        by=['uid', 'date'],
                                        as_index=False)

In [15]: total_purchases = purchase_data_agg.purchase.sum()

In [16]: total_purchases.purchase = np.where(
                                    np.isnan(total_purchases.purchase),
                                    0, total_purchases.purchase)

In [17]:total_purchases.purchase.mean()

Out[17]: 0.0346

In [18]: min(total_purchases.purchase)

Out[18]: 0.0

In [19]: max(total_purchases.purchase)

Out[19]: 3.0
```

Randomness of Experimental Units

Designing Your A/B Test

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

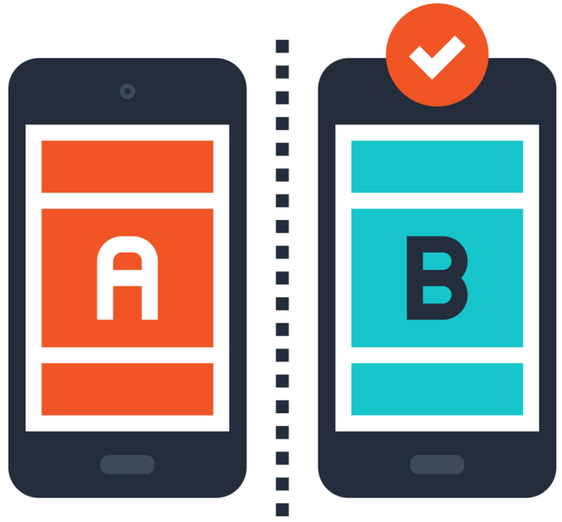
Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

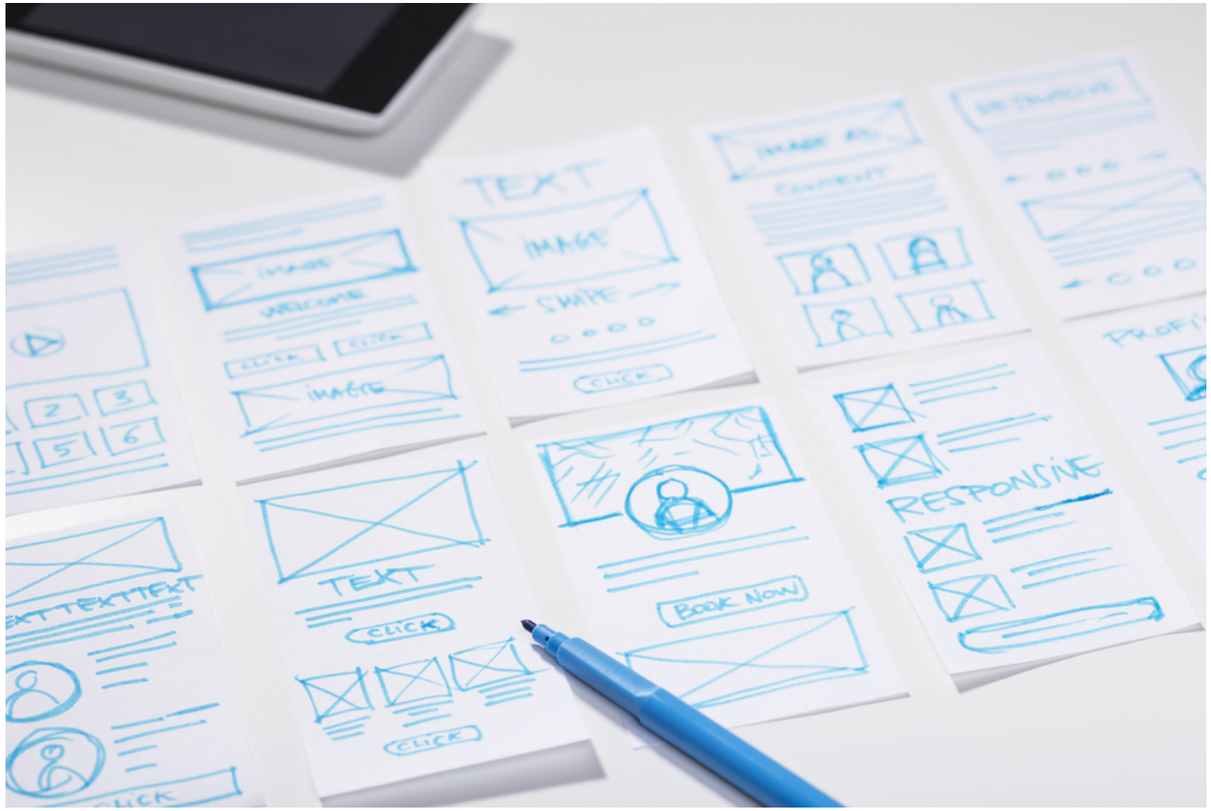
Preparing to run an A/B test

Ryan Grossman
Data Scientist, EDO

Paywall A/B Test Variants



Main Concerns in Designing a Test



Test Sensitivity

Evaluating Different Sensitivities

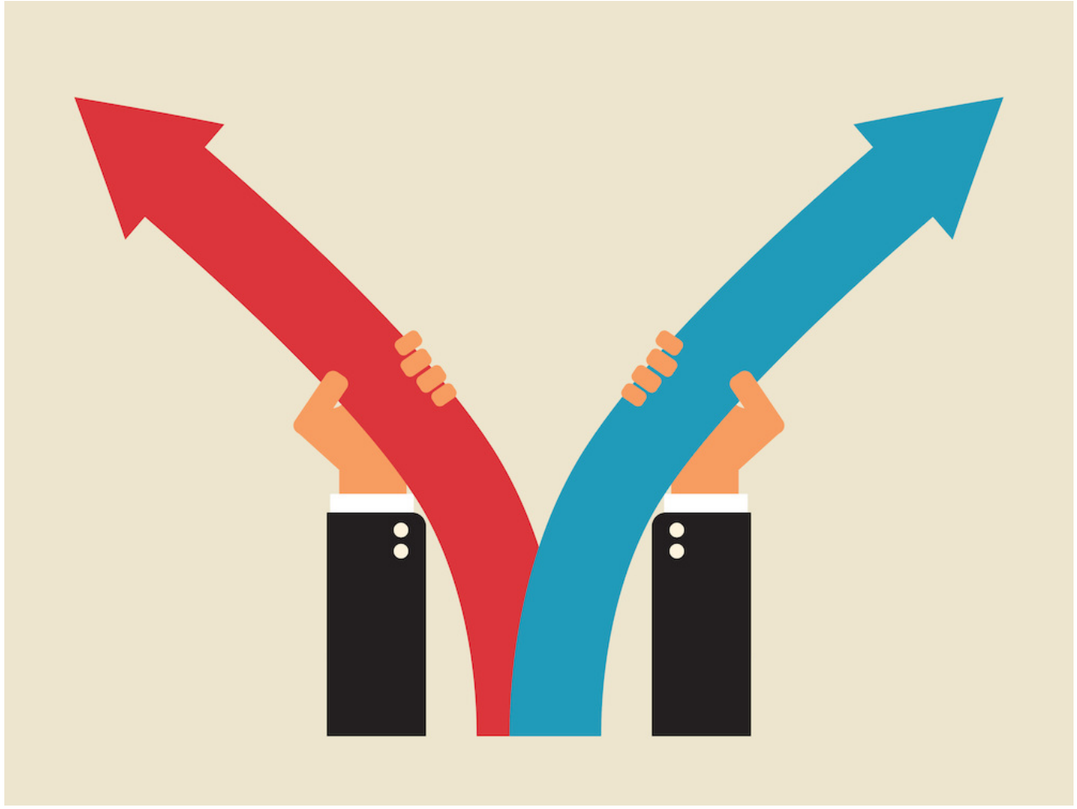
Finding Revenue Per User

```
In [1]: purchase_data = demographics_data.merge(paywall_views,
                                                how='left', on=['uid'])

In [2]: purchase_data_agg = purchase_data.groupby(by=['uid'],
                                                  as_index=False)

In [3]: total_revenue = purchase_data_agg.price.sum()

In [4]: total_revenue.price = np.where(np.isnan(total_revenue.price),
                                       0, total_revenue.price)

In [5]: avg_revenue = total_revenue.price.mean()

In [6]: avg_revenue

Out[6]: 16.161
```

Evaluating Our Sensitivities

```
In [7]: one_pct_lift = avg_revenue * 1.01

In [8]: one_pct_lift

Out[8]: 16.322839545454478

In [9]: ten_pct_lift = avg_revenue * 1.1

In [10]: ten_pct_lift

Out[10]: 17.77

In [11]: twenty_pct_lift = avg_revenue * 1.2

In [12]: twenty_pct_lift

Out [12]: 19.393
```

Understanding Variability in Our Data



Standard Deviation

```
In [13]: revenue_variation = total_revenue.price.std()

In [14]: revenue_variation

Out[14]: 17.520
```

Variability of Revenue Per User

```
In [15]: revenue_variation / avg_revenue

Out[15]: 1.084
```

Variability of Purchases Per User

```
In [16]: total_purchases = purchase_data_agg.purchase.sum()

In [17]: total_purchases.purchase = np.where(
                                        np.isnan(total_purchases.purchase),
                                        0, total_purchases.purchase)

In [18]: avg_purchases = total_purchases.purchase.mean()

In [19]: avg_purchases

Out[19]: 3.15

In [20]: purchase_variation = total_purchases.purchase.std()

In [21]: purchase_variation

Out[21]: 2.68

In [22]: purchase_variation / avg_purchases

Out [22]: 0.850
```

Choosing our Experimental Unit & Response Variable

Choosing our Experimental Unit & Response Variable

```
In [23]: purchase_data = demographics_data.merge(paywall_views,
                                how='inner', on=['uid'])

In [24]: purchase_data_agg = purchase_data.groupby(by=['uid'],
                                                   as_index=False)

In [25]: conversion_rate = (sum(purchase_data.purchase) /
            purchase_data.purchase.count())

In [26]: conversion_rate

Out[26]: 0.347
```

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

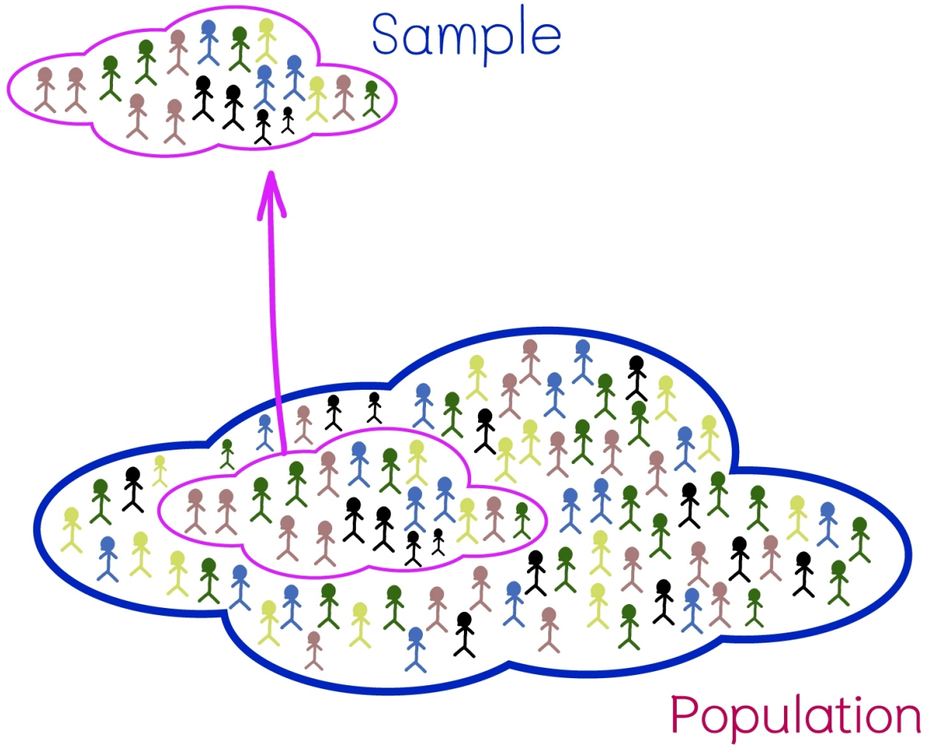
Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Calculating Sample Sizes

Ryan Grossman
Data Scientist, EDO

Calculating Sample Sizes



Null Hypothesis

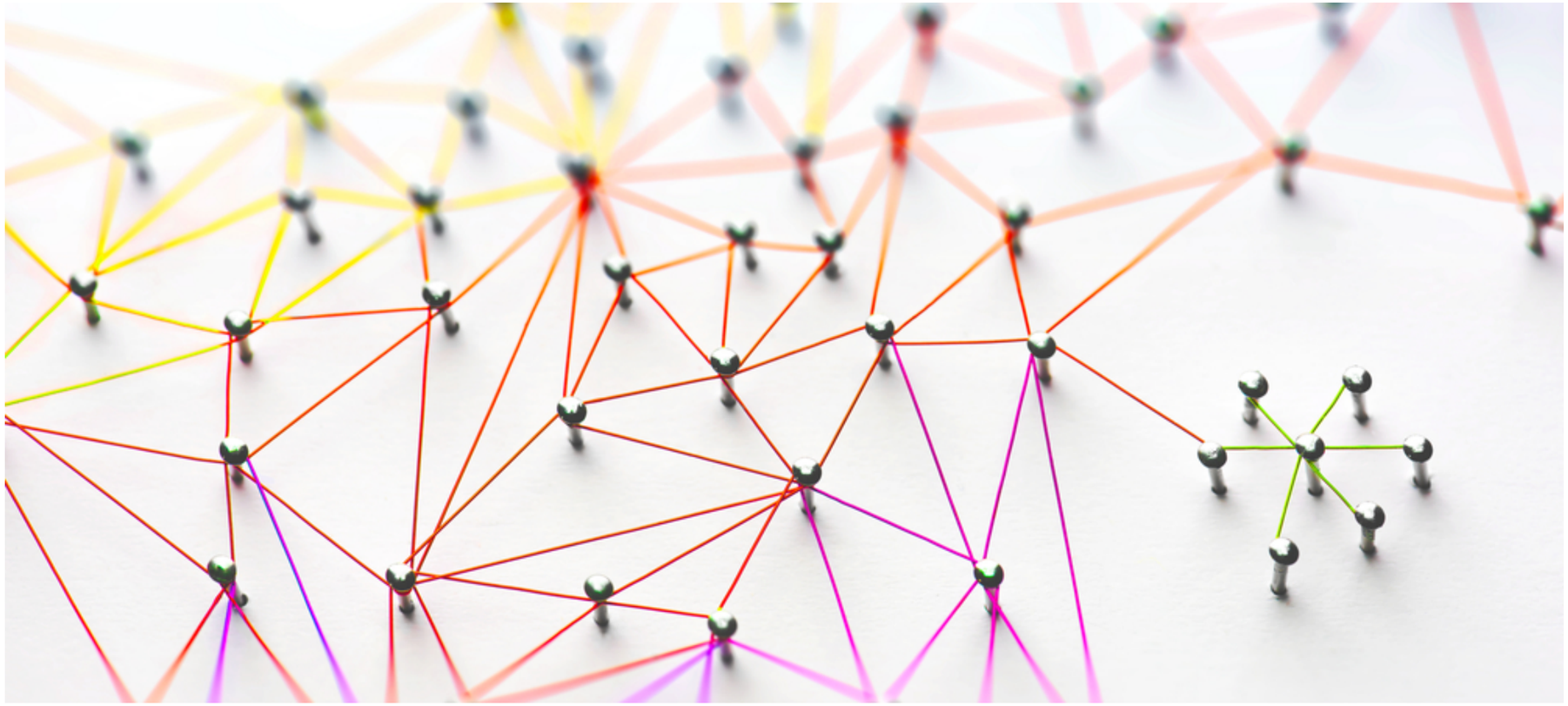
Types of Error

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

Confidence Level

Statistical Power

Connecting the Different Components



Power Formula

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

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

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

$$diff = |p_1 - p_2|$$

$$\bar{p} = \frac{(p_1 + p_2)}{2}$$

$$v_1 = p_1 \times (1 - p_1),\ v_2 = p_2 \times (1 - p_2)$$

$$\bar{v} = \bar{p} \times (1 - \bar{p})$$

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

Sample Size Function

```
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

    v1 = p1 * (1 - p1)
    v2 = p2 * (1 - p2)
    bv = bp * (1 - bp)
    power_part_one = stats.norm.cdf((n**0.5 * diff - qu * (2 * bv)**0.5)/
                            (v1 + v2)**0.5)
    power_part_two = 1 - stats.norm.cdf((n**0.5 * diff + qu * (2 * bv)**0.5)/
                            (v1 + v2)**0.5)

    power = power_part_one + power_part_two
    return(power)
```

```
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

```
In [3]: sample_size_per_group =  get_sample_size(0.8,
            conversion_rate, conversion_rate * 1.1, 0.95)

In [4]: sample_size_per_group

Out[4]: 45788
```

Generalness of this Function

Decreasing the Sample Size

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Let's practice!

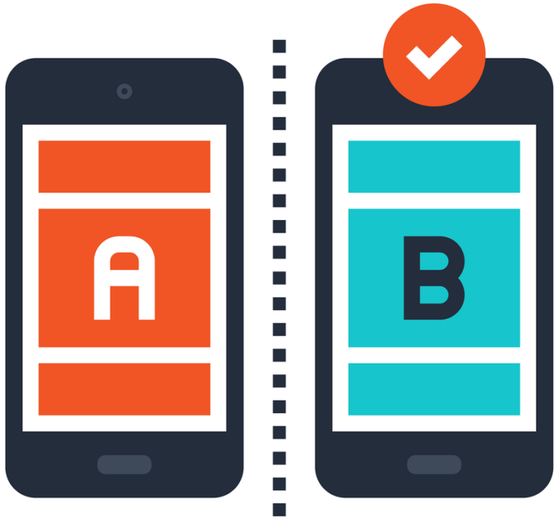
CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Analyzing the A/B test results

Ryan Grossman
Data Scientist, EDO

Analyzing A/B Test Results

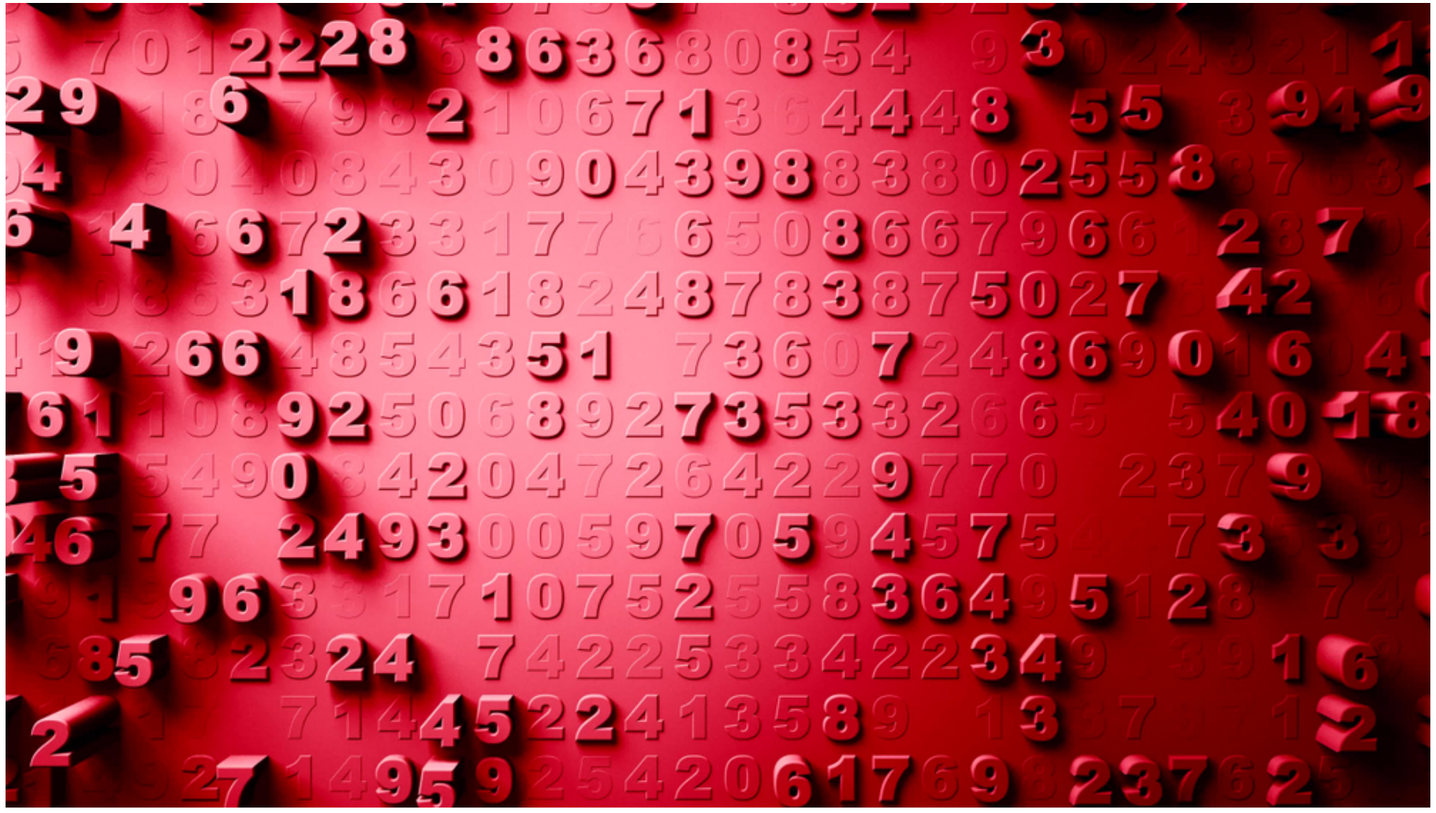
Evaluating our Test



Test Results

```
In [1]: test_demographics = pd.read_csv('test_demographics.csv`)

In [2]: test_results = pd.read_csv('ab_test_results.csv')

In [3]: test_results.date = pd.to_datetime(test_results.date)

In [4]: test_results.head(n=5)

Out[4]:
uid                 date    purchase    sku    price    group
90554036.0    2018-02-27 14:22:12    0    NaN    NaN    C
90554036.0    2018-02-28 08:58:13    0    NaN    NaN    C
90554036.0    2018-03-01 09:21:18    0    NaN    NaN    C
90554036.0    2018-03-02 10:14:30    0    NaN    NaN    C
90554036.0    2018-03-03 13:29:45    0    NaN    NaN    C
```

Confirming Test Results



Confirming Test Results

```
In [5]: test_results_grpd = test_results.groupby(by=['group'],
                                                 as_index=False)

In [6]: test_results_grpd.uid.count()

Out[6]:
    group    uid
0     C      48236
1     V      49867
```

Confirming Test Results

```
In [7]: test_results_demo = test_results.merge(test_demo,
                                  how='inner', on='uid')

In [8]: test_results_grpd = test_results_demo.groupby(by=
                                  ['country','gender', 'device', 'group' ],
                                  as_index=False)

In [9]: test_results_grpd.uid.count()

Out[9]:
country    gender    device    group    uid
BRA        F         and       C        5070
BRA        F         and       V        4136
BRA        F         iOS       C        3359
BRA        F         iOS       V        2817
BRA        M         and       C        3562
BRA        M         and       V        3673
BRA        M         iOS       C        2940
BRA        M         iOS       V        3109
CAN        F         and       C        747
CAN        F         and       V        806
CAN        F         iOS       C        447
```

Finding The Test & Control Group Conversion Rates

```
In [10]: test_results_grpd = test_results_demo.groupby(
                                  by=['group'], as_index=False)

In [11]: test_results_summary = test_results_grpd.agg(
                                      {'purchase': ['count', 'sum']})

In [12]: test_results_summary

Out[12]:
    group    purchase
count    sum
0    C     48236    1657
1    V     49867    2094

In [13]: test_results_summary['conv'] = (
                test_results_summary.purchase['sum'] /
                test_results_summary.purchase['count'])

In [14]: test_results_summary

Out[14]:
    group    purchase    conv
count    sum
0    C     48236    1657    0.034351
1    V     49867   2094     0.041984
```


STATISTICAL SIGNIFICANCE

p-Values

p-value:

- Probability under the Null Hypothesis of obtaining a result as or more extreme than the one observed.
- Represents a measure of the evidence against retaining the Null Hypothesis.

Interpreting a p-Value

p-value	Conclusion
< 0.01	very strong evidence against the Null Hypothesis
0.01 - 0.05	strong evidence against the Null Hypothesis
0.05 - 0.10	very weak evidence against the Null Hypothesis
> 0.1	small to no evidence against the Null Hypothesis

Next Steps

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Let's practice!

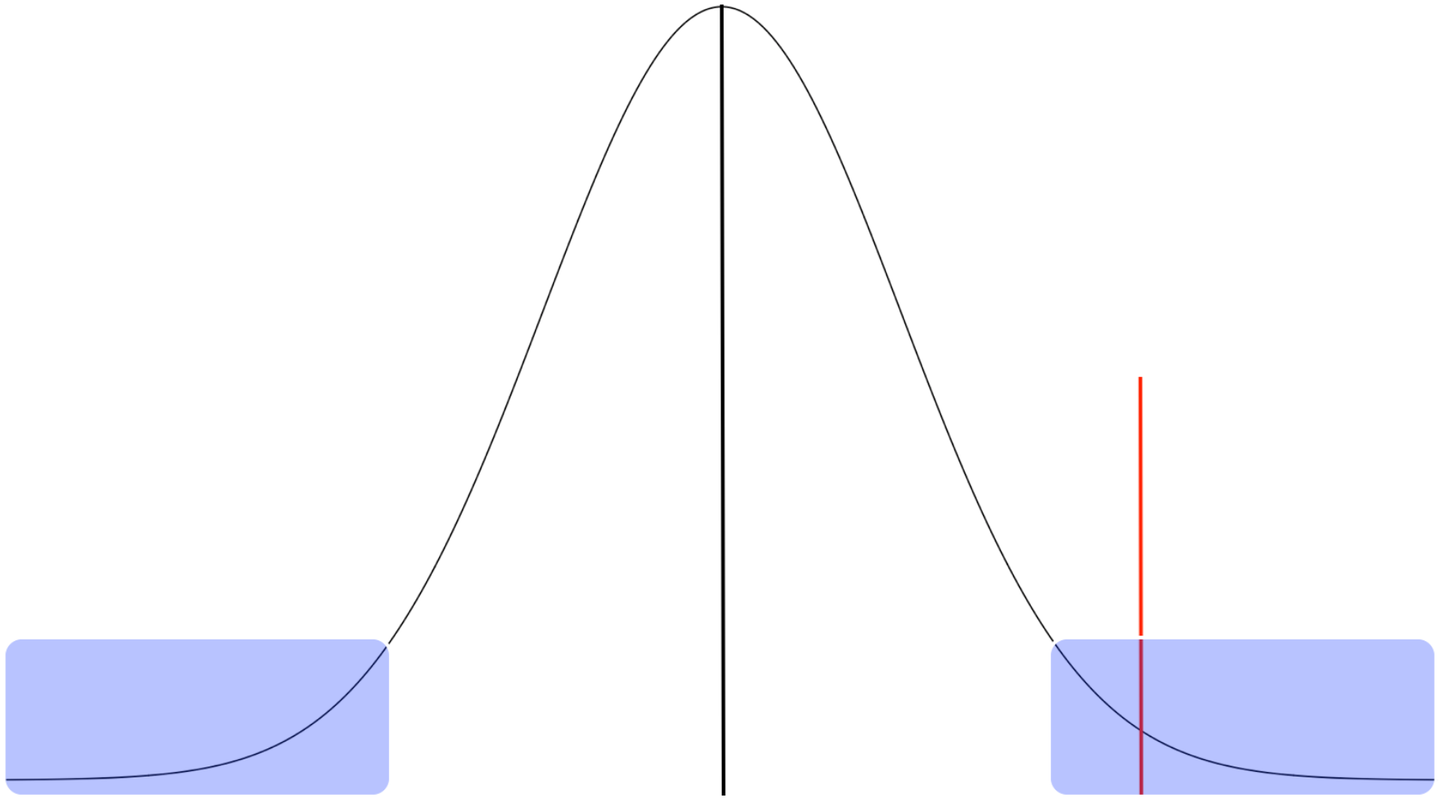
CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Understanding statistical significance

Ryan Grossman
Data Scientist, EDO

Next Steps In Our Analysis

Revisiting Statistical Significance



p-value Function

```
def get_pvalue(con_conv, test_conv,con_size,  test_size,):
    lift =  - abs(test_conv - con_conv)

    scale_one = con_conv * (1 - con_conv) * (1 / con_size)
    scale_two = test_conv * (1 - test_conv) * (1 / test_size)
    scale_val = (scale_one + scale_two)**0.5

    p_value = 2 * stats.norm.cdf(lift, loc = 0, scale = scale_val )

    return p_value
```

Calculating our p-value

```
In [1]: con_conv = 0.034351
In [2]: test_conv = 0.041984
In [3]: con_size = 48236
In [4]: test_size = 49867

In [5]: p_value = get_pvalue(con_conv, test_conv, con_size, test_size)

In [6]: p_value

Out[6]: 4.2572974855869089e-10
```

Finding the Test Power

```
In [7]: get_power(test_size, con_conv, test_conv, 0.95)

Out[7]: 0.999992594137228 19
```


CONFIDENCE
INTERVAL

Confidence Intervals

Confidence Interval

- Provides contextualizaton of the estimation process.
- The conversion rate is a fixed quantitiy, the estimation is what is variable.

Confidence Intervals

Two Sided Confidence Interval

- $\mu \pm \Phi \left(\alpha + \frac{1-\alpha}{2}\right) * \sigma$
- μ: Estimated Mean
- σ: Estimated Standard Deviation
- α: Desired Confidence Interval Width

Calculating Confidence Intervals

```
def get_ci(lift, alpha, sd):
  val = abs(stats.norm.ppf((1 - alpha)/2))

  lwr_bnd = lift - val * sd
  upr_bnd = lift + val * sd

  return_val = (lwr_bnd, upr_bnd)
  return(return_val)
```

Calculating Confidence Intervals

```
In [8]: lift = test_conv - con_conv

In [9]: sd = ((test_conv * (1 - test_conv)) / test_size +
              (con_conv * (1 - con_conv)) / con_size)**0.5

In [10]: get_ci(lift, 0.95, sd)

Out[10]: (0.00523714629485782 72, 0.0100288537051421 75)
```

Next Steps

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

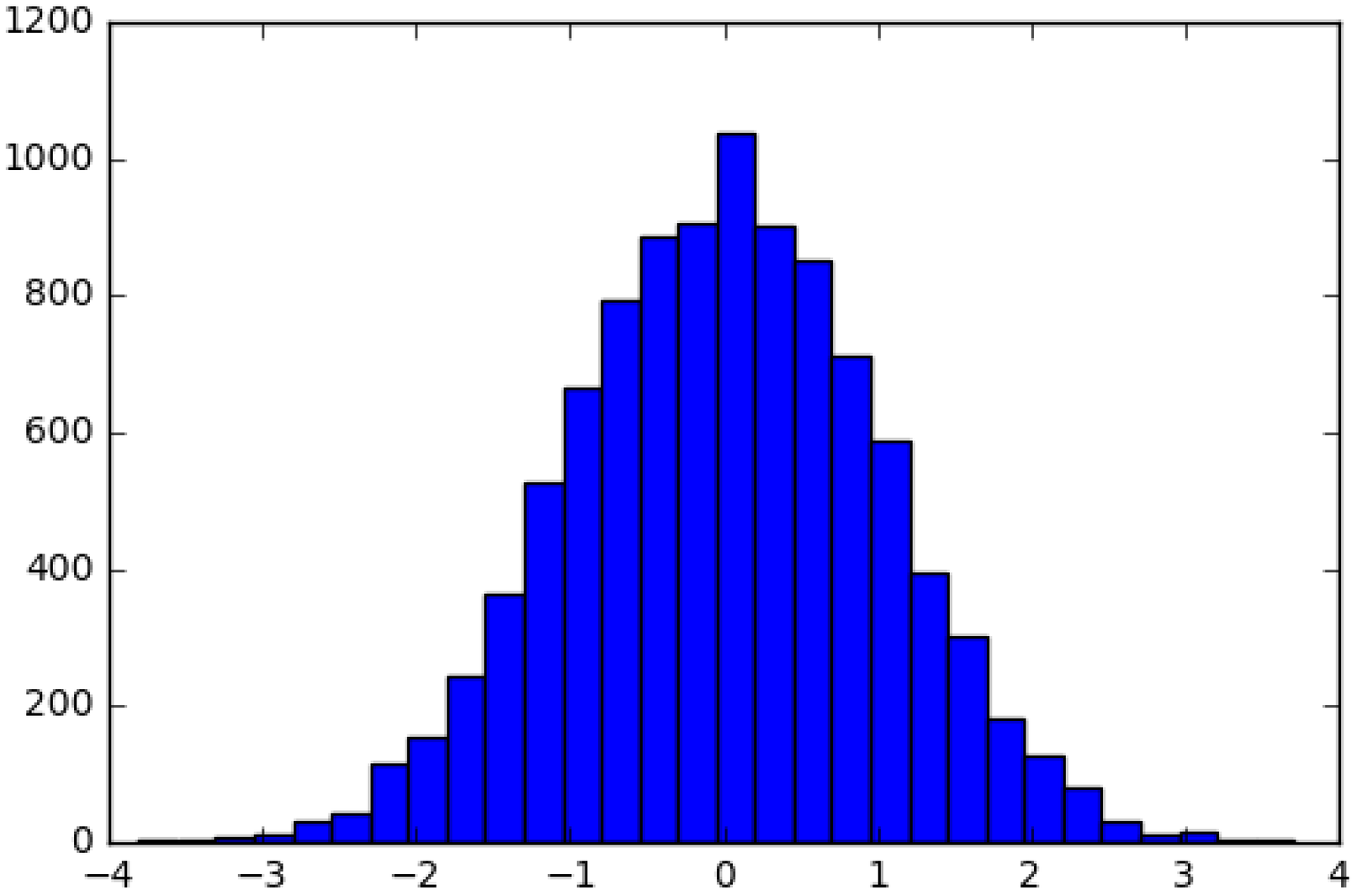
Interpreting your test results

Ryan Grossman
Data Scientist, EDO

Communicating Your Test Results

	Test Group	Control Group
Sample Size	7030	6970
Run Time	2 Weeks	2 Weeks
Mean	3.12	2.69
Variance	3.20	2.64
Estimated Lift: 0.56 *		
Confidence Intervel 0.56 ± 0.4		

**** Significant at the 0.05 Level***

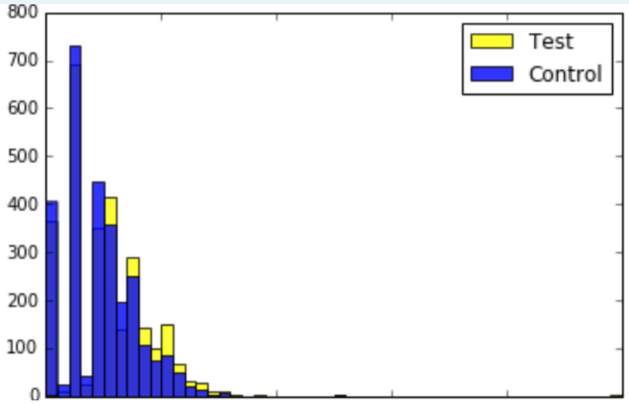

1200
1000
800
600
400
200
0
−4
−3
−2
−1
0
1
2
3
4

Generating Histograms - Data

```
In [1]: test_results_rollup.head(n=10)

Out[1]:
    uid      group    purchase
11128497.0    V        0.000000
11145206.0    V        0.050000
11163353.0    C        0.150000
11215368.0    C        0.000000
11248473.0    C        0.157895
11258429.0    V        0.086957
11271484.0    C        0.071429
11298958.0    V        0.157895
11325422.0    C        0.045455
11340821.0    C        0.040000
```

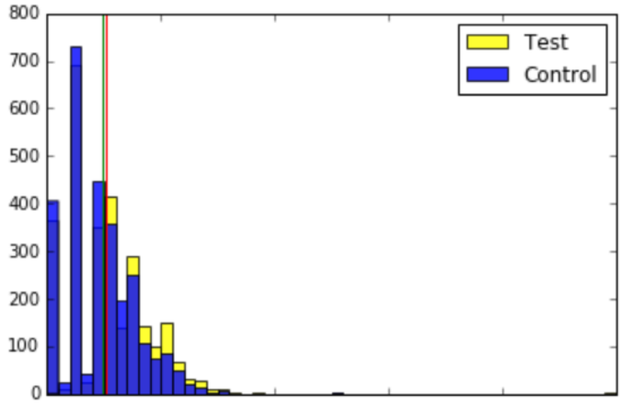
Generating Histograms - Code

```
In [2]: variant_results_rollup = test_results_rollup[
            test_results_rollup.group == 'V']

In [3]: control_results_rollup = test_results_rollup[
            test_results_rollup.group == 'C']

In [4]: plt.hist(variant_results_rollup['purchase'],
            color = 'yellow', alpha = 0.8, bins =50, label = 'Test')

In [5]: plt.hist(control_results_rollup['purchase'],
            color = 'blue', alpha = 0.8, bins = 50, label = 'Control')

In [6]: plt.legend(loc='upper right')

In [7]: plt.show()
```



Adding Lines & Annotations

```
In [8]: plt.axvline(x = np.mean(variant_results_rollup.purchase), color = 'red')

In [9]: plt.axvline(x= np.mean(test_results_rollup.purchase), color = 'green')

In [10]: plt.show()
```



Plotting the Distribution

```
In [11]: mean_control = 0.090965
In [12]: mean_test = 0.102005
In [13]: var_control = (mean_control * (1 - mean_control)) / 58583
In [14]: var_test = (mean_test * (1 - mean_test)) / 56350

In [15]: control_line = np.linspace(-3 * var_control**0.5 +
                mean_control, 3 * var_control**0.5 +
                mean_control, 100)

In [16]: test_line = np.linspace(-3 * var_test**0.5 +
              mean_test, 3 * var_test**0.5
              + mean_test, 100)
```

Plotting the Distribution

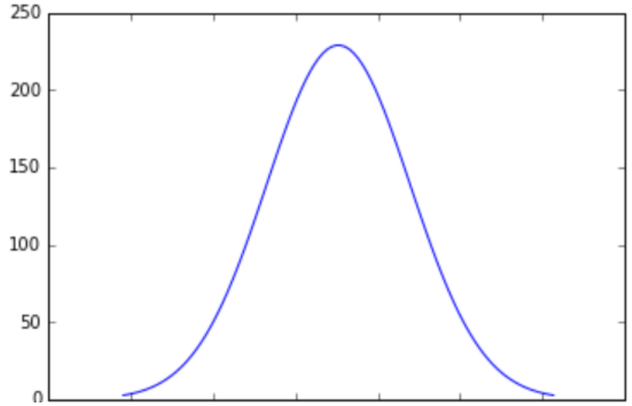
```
In [17]: plt.plot(control_line,mlab.normpdf(control_line,
                  mean_control, var_control**0.5))

In [18]: plt.plot(test_line,mlab.normpdf(test_line,
                  mean_test, var_test**0.5))

In [19] plt.show()
```

Plotting the Difference of Distributions

```
In [20]: lift = mean_test - mean_control

In [21]: var = var_test + var_control
```

Plotting the Difference of Distributions

```
In [22]: diff_line = np.linspace(-3 * var**0.5 +
             lift, 3 * var**0.5
             + lift, 100)

In [23]: plt.plot(diff_line,mlab.normpdf(diff_line,
                 lift, var**0.5))

In [24] plt.show()
```



Plotting the Confidence Interval

```
In [25]: section = np.arange(0.007624, 0.01445 , 1/10000)

In [26]: plt.fill_between(section,
                          mlab.normpdf(section,
                          lift, var**0.5))

In [27]: plt.plot(diff_line,mlab.normpdf(diff_line,
                         lift, var**0.5))

In [28]: plt.show()
```

Plotting the Confidence Interval

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Finale

Ryan Grossman
Data Scientist, EDO

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Let's practice!