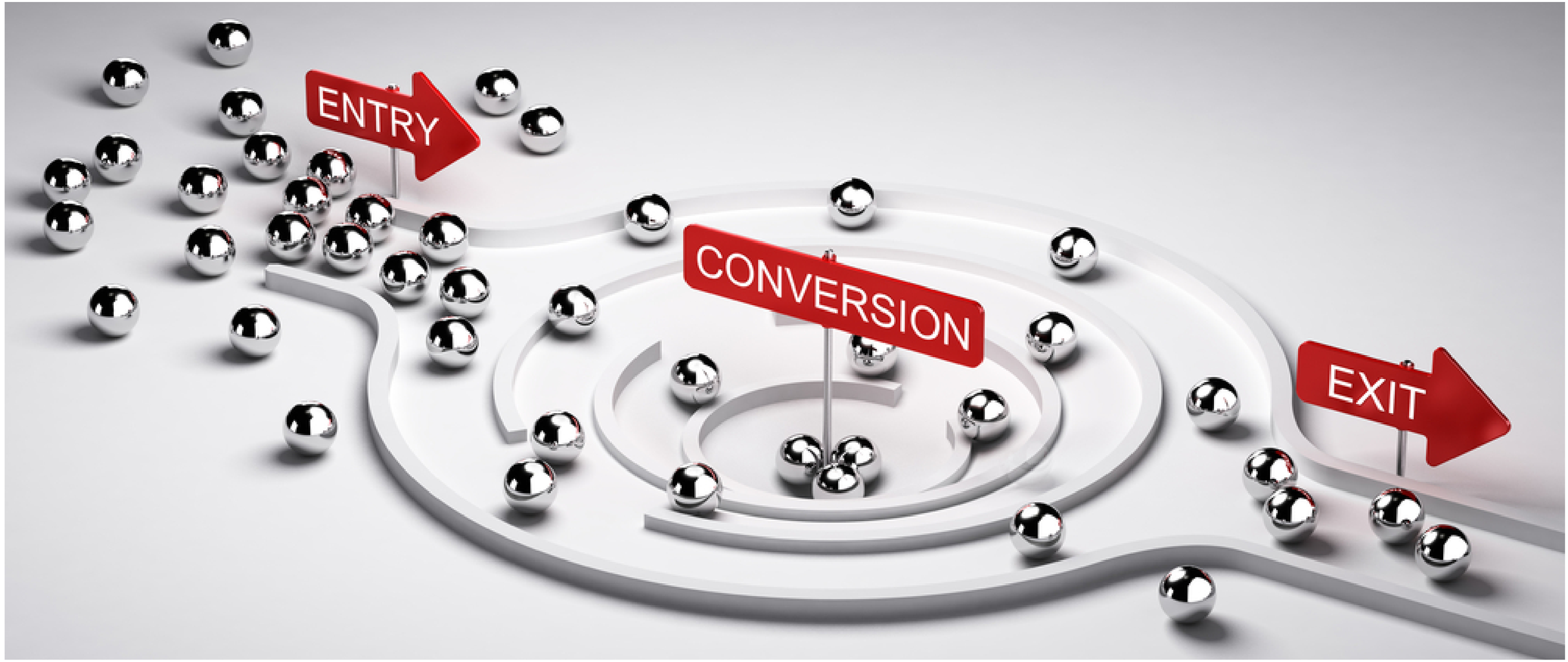
Working with time series data in pandas

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

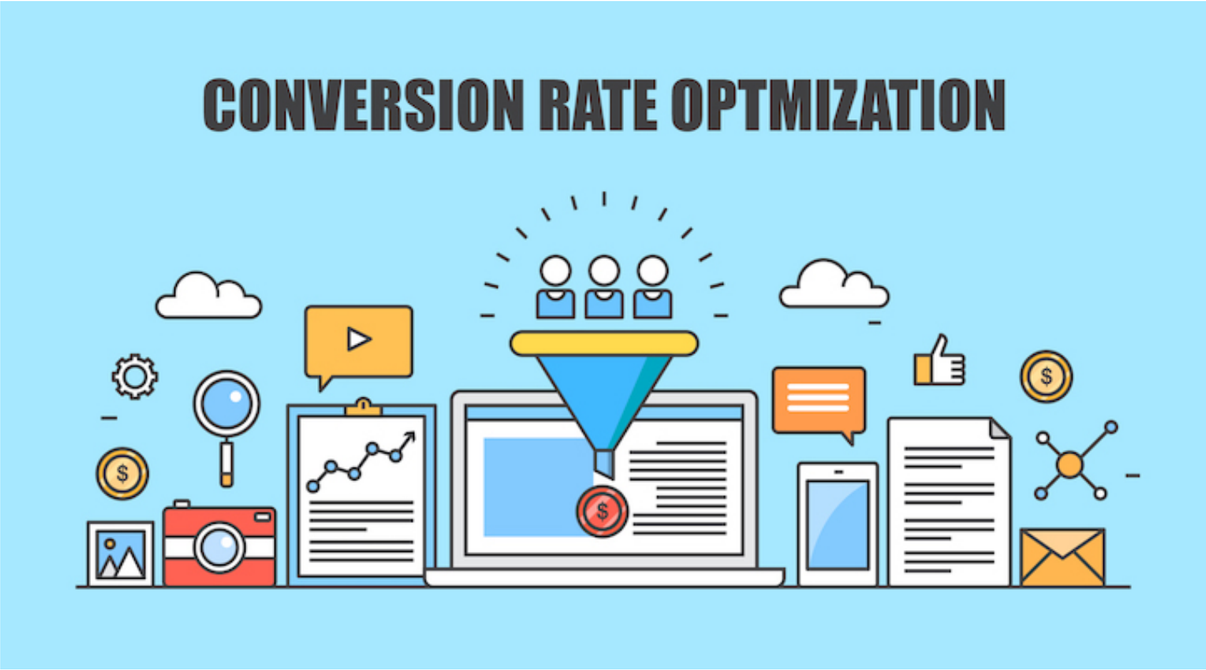
Ryan Grossman
Data Scientist, EDO

Exploratory Data Analysis

Dates & Times



Week Two Conversion Rate



Using the Timedelta Class

```
current_date = pd.to_datetime('2018-03-17')

max_lapse_date = current_date - timedelta(days=14)

conv_sub_data = sub_data_demo[sub_data_demo.lapse_date < max_lapse_date]
```

Date Differences

```
sub_time = (conv_sub_data.subscription_date
            - conv_sub_data.lapse_date)

conv_sub_data['sub_time'] = sub_time
```

Date Components

```
conv_sub_data['sub_time'] = conv_sub_data.sub_time.dt.days
```

Conversion Rate Calculation

```
conv_base = conv_sub_data[(conv_sub_data.sub_time.notnull()) | (conv_sub_data.sub_time > 7)]
total_users = len(conv_base)
```

```
2086
```

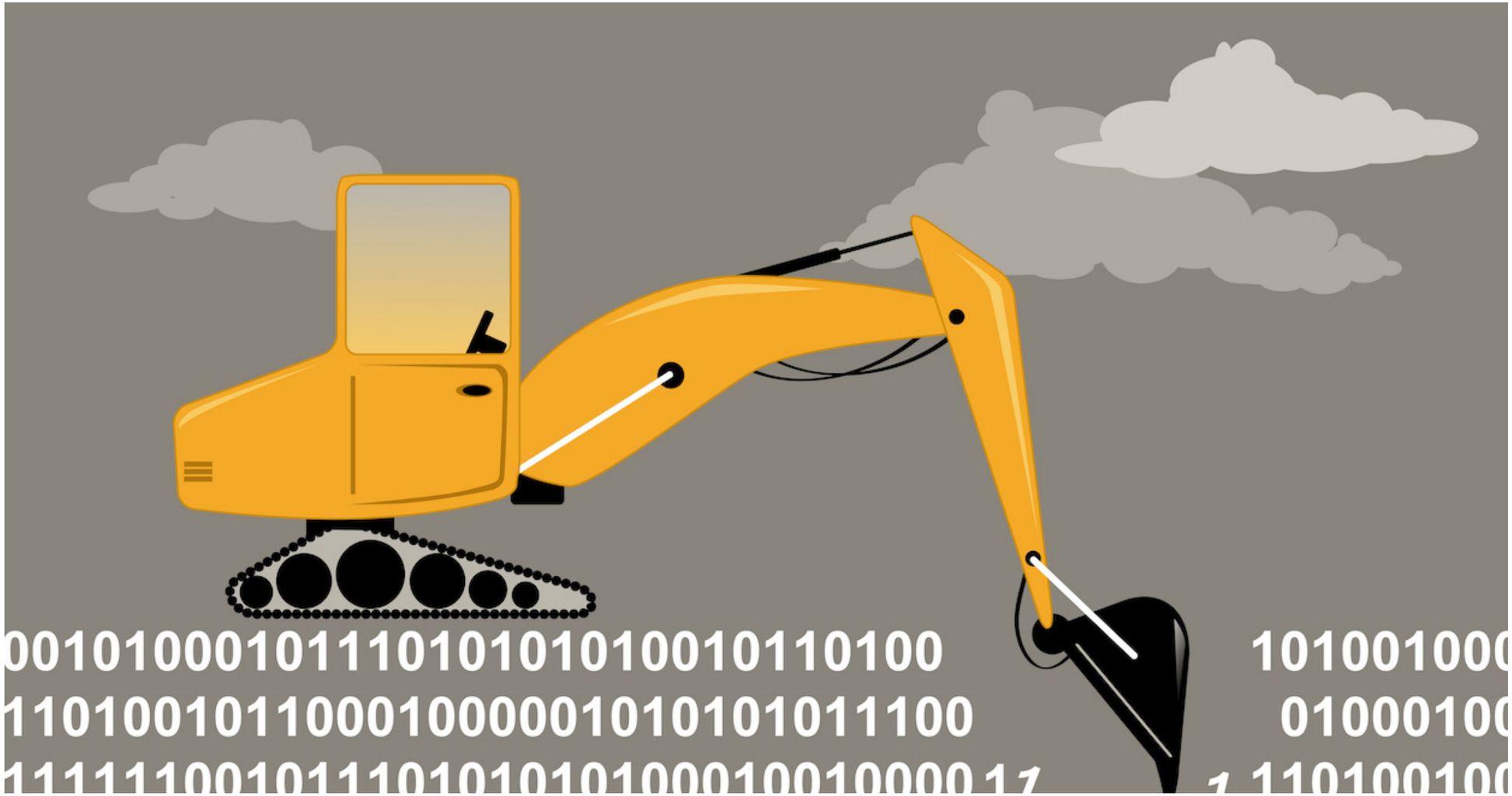
```
total_subs = np.where(conv_sub_data.sub_time.notnull() & (conv_base.sub_time <= 14), 1, 0)
total_subs = sum(total_subs)
```

```
20
```

```
conversion_rate = total_subs / total_users
```

```
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,...)
```

```
customer_demographics = pd.read_csv('customer_demographics.csv',
    parse_dates=True,
    infer_datetime_format=True)
```

```
    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
```

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"`

Let's practice!

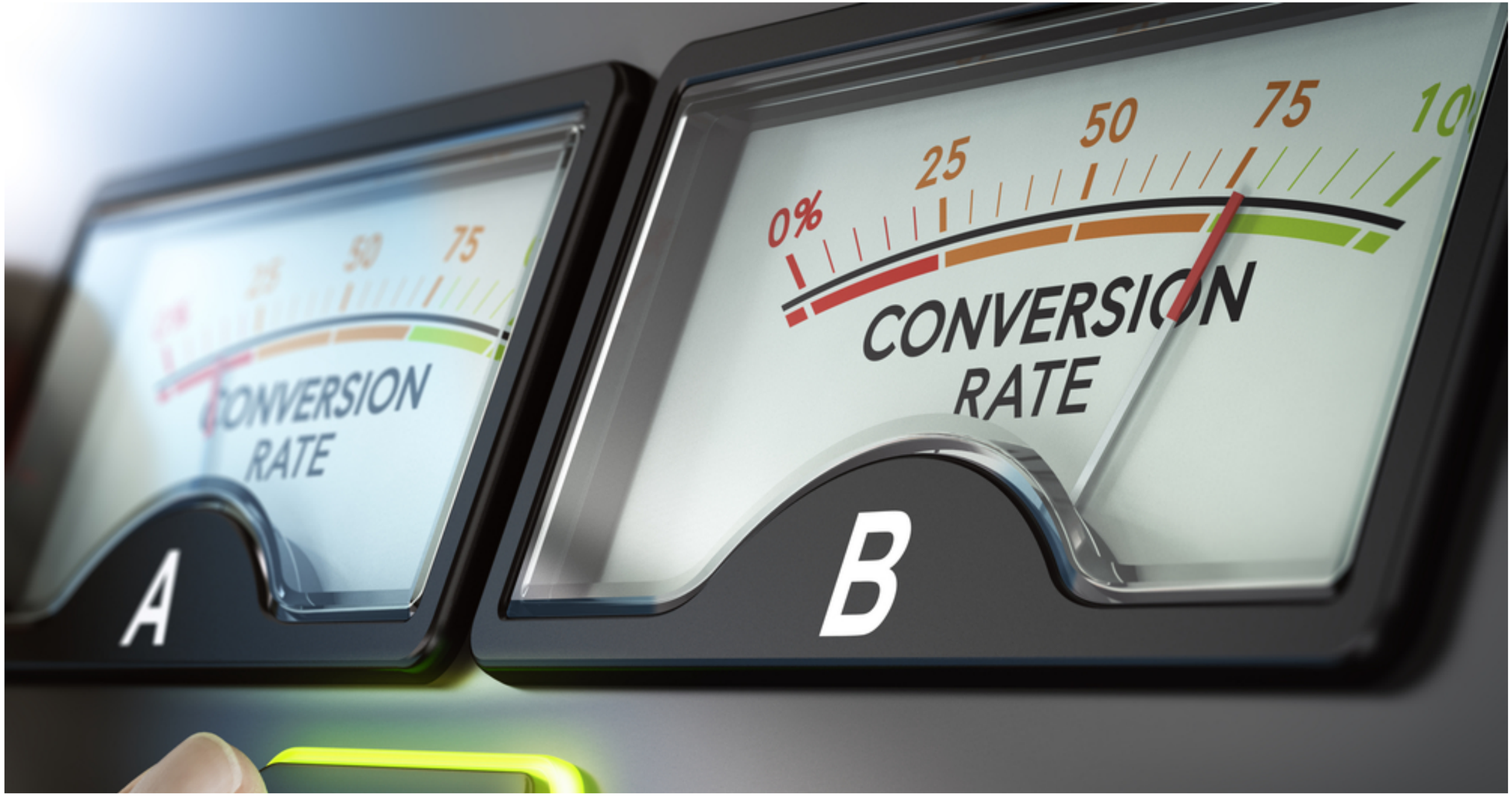
CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Creating time series graphs With matplotlib

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Ryan Grossman
Data Scientist, EDO

Conversion Rate Over Time



Monitoring The Impact of Changes

Conversion Rate by Day

```
current_date = pd.to_datetime('2018-03-17')

max_lapse_date = current_date - timedelta(days=7)

conv_sub_data = sub_data_demo[sub_data_demo.lapse_date
                              < max_lapse_date]

sub_time = (conv_sub_data.subscription_date -
                         conv_sub_data.lapse_date).dt.days

conv_sub_data['sub_time'] = sub_time
```

Conversion Rate by Day

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

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

conversion_data.columns = conversion_data.columns.droplevel(
                                            level=1)

conversion_data.head()
```

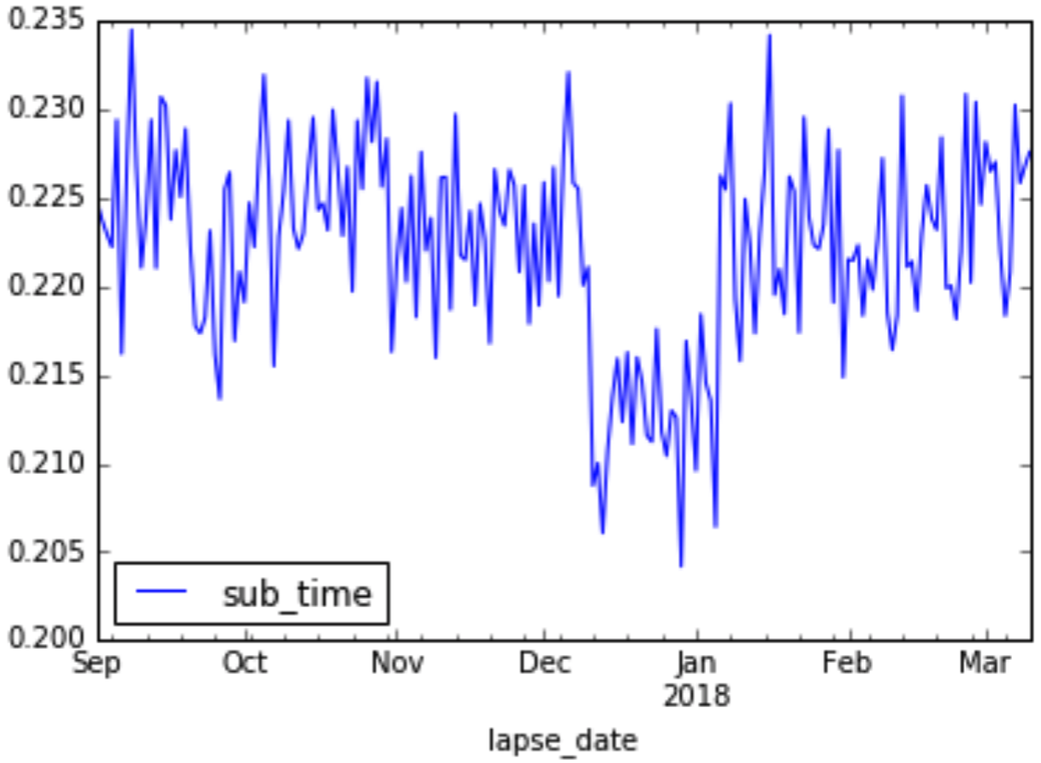
```
    lapse_date    sub_time
0    2017-09-01    0.224775
1    2017-09-02    0.223749
2    2017-09-03    0.222948
3    2017-09-04    0.222222
4    2017-09-05    0.229401
```

Plotting Daily Conversion Rate

```
conversion_data.lapse_date =
            pd.to_datetime(conversion_data.lapse_date)

conversion_data.plot(x='lapse_date', y='sub_time')
```

Plotting Daily Conversion Rate

```
plt.show()
```



Trends in Different Cohorts

```
conversion_data.head()
```

```
    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()`

```
reformatted_cntry_data =pd.pivot_table(conversion_data, ...)

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

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

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

`.pivot_table()`

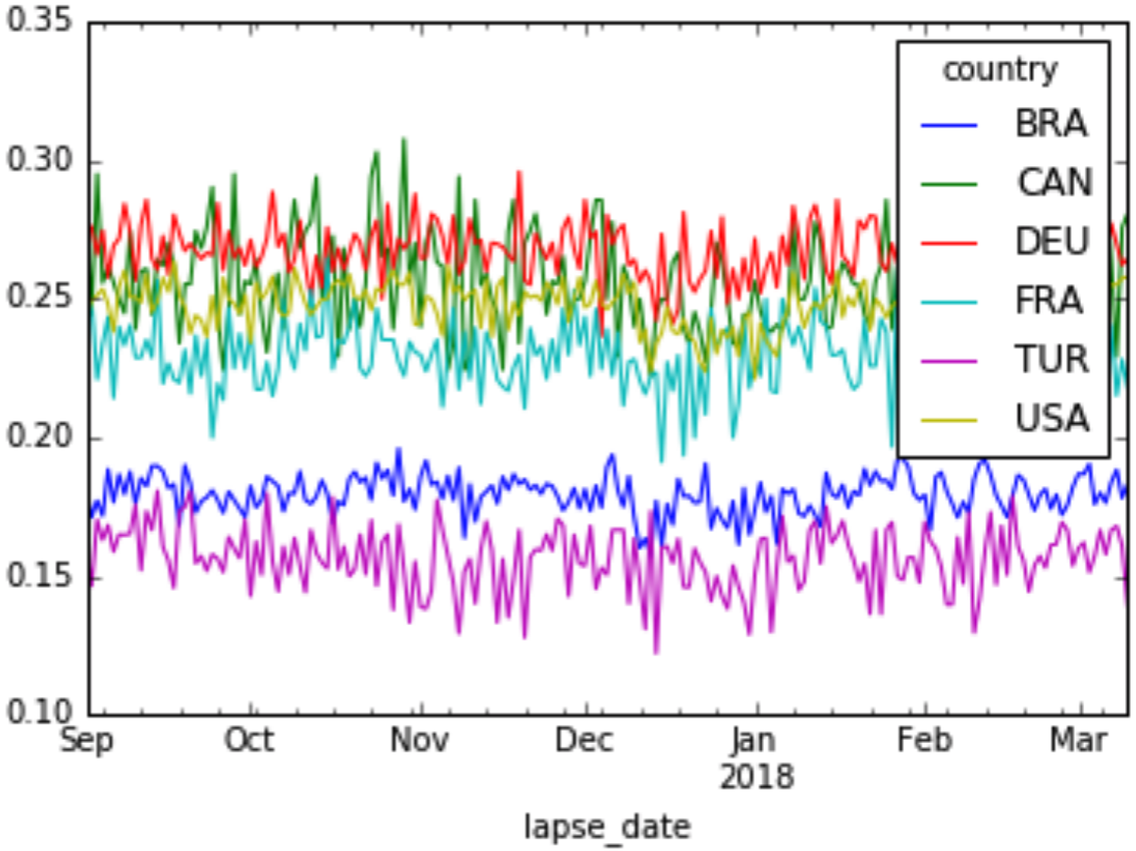
```
reformatted_cntry_data.columns
    reformatted_cntry_data.columns.droplevel(level=[0])

reformatted_cntry_data.reset_index(inplace=True)

reformatted_cntry_data.head()
```

```
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

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

plt.show()
```

Plotting Trends in Different Cohorts



Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Understanding and visualizing trends in customer data

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Ryan Grossman
Data Scientist, EDO

Further Techniques for Uncovering Trends

Subscribers Per Day

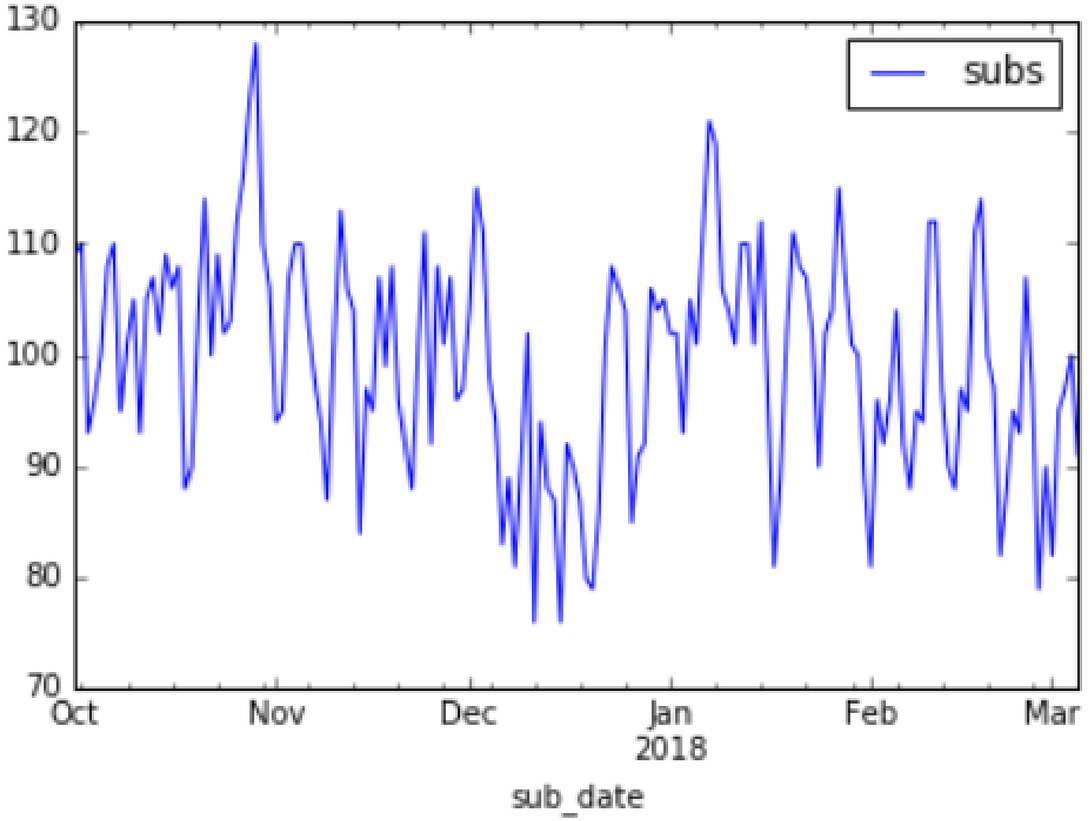
```
usa_subscriptions = pd.read_csv('usa_subscribers.csv',
                                parse_dates=True,
                                infer_datetime_format=True)
usa_subscriptions['sub_day'] = (usa_subscriptions.sub_date -
                                    usa_subscriptions.lapse_date).dt.days
usa_subscriptions = usa_subscriptions[usa_subscriptions.sub_day <= 7]
usa_subscriptions = usa_subscriptions.groupby(by=['sub_date'], as_index = False)
usa_subscriptions = usa_subscriptions.agg({'subs': ['sum']})
usa_subscriptions.columns = usa_subscriptions.columns.droplevel(level=[1])
usa_subscriptions.head()
```

```
    sub_date        subs
0    2016-09-02        37
1    2016-09-03        50
2    2016-09-04        59
```

Subscribers Per Day

```
usa_subscriptions.plot(x='sub_date', y='subs')

plt.show()
```

Seasonality



Correcting for Seasonality

Trailing Averages

Calculating Trailing Averages

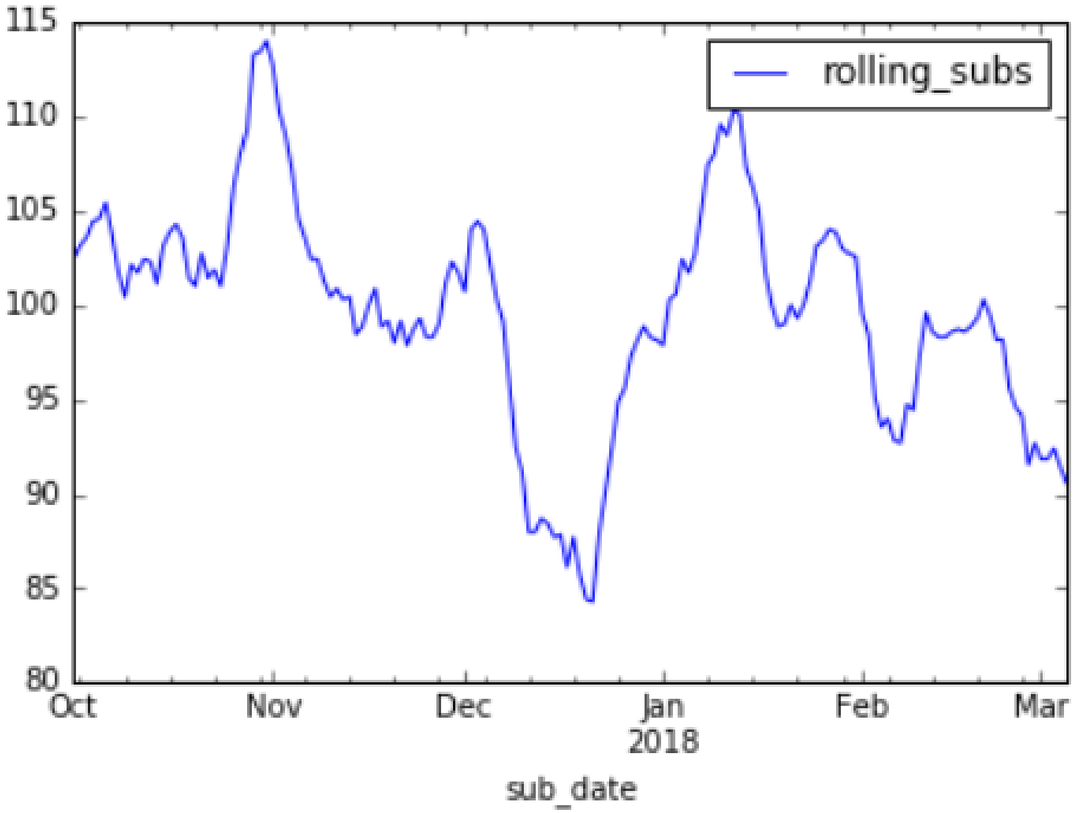
```
rolling_subs =  usa_subscriptions.subs.rolling(...)

rolling_subs =  usa_subscriptions.subs.rolling(window=7, ...)

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

Calculating Trailing Averages

```
rolling_subs = rolling_subs.mean()

usa_subscriptions['rolling_subs'] = rolling_subs

usa_subscriptions.tail()
```

```
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

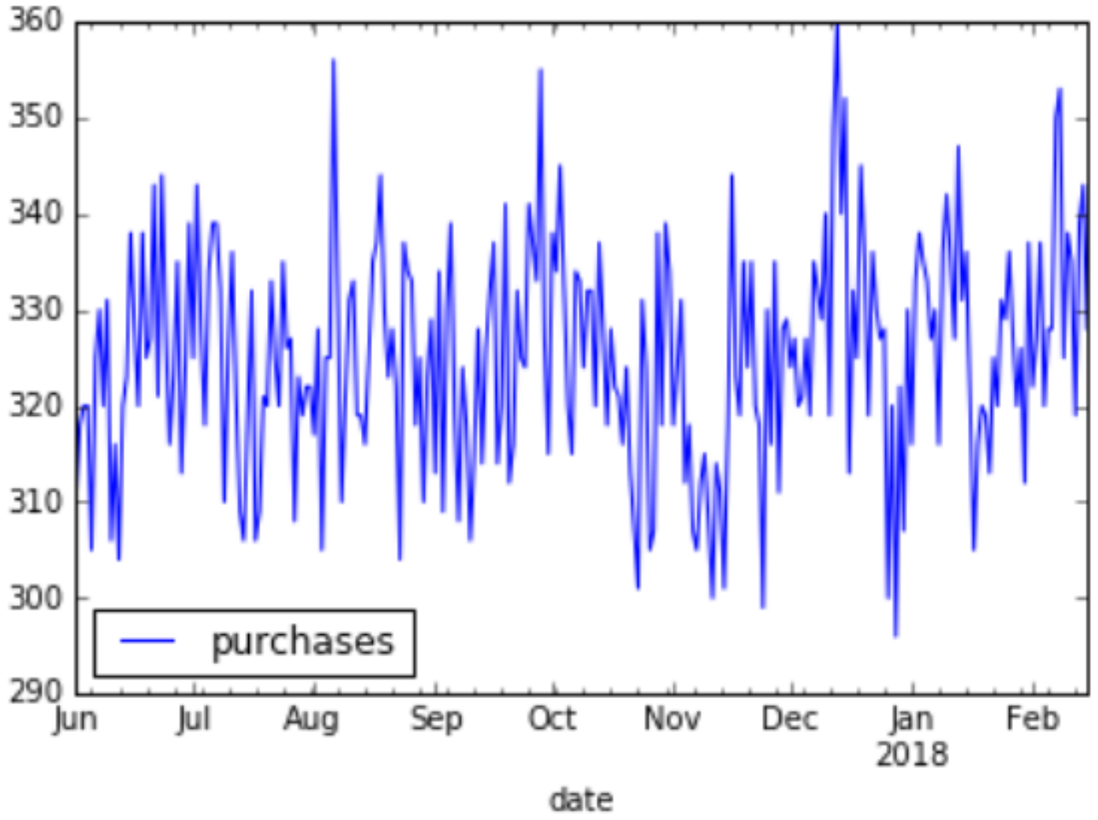


Noisy Data

Noisy Data

```
high_sku_purchases = pd.read_csv('high_sku_purchases.csv',
                                 parse_dates=True,
                                 infer_datetime_format=True)

high_sku_purchases.plot(x='date', y='purchases')

plt.show()
```

Exponential Moving Average

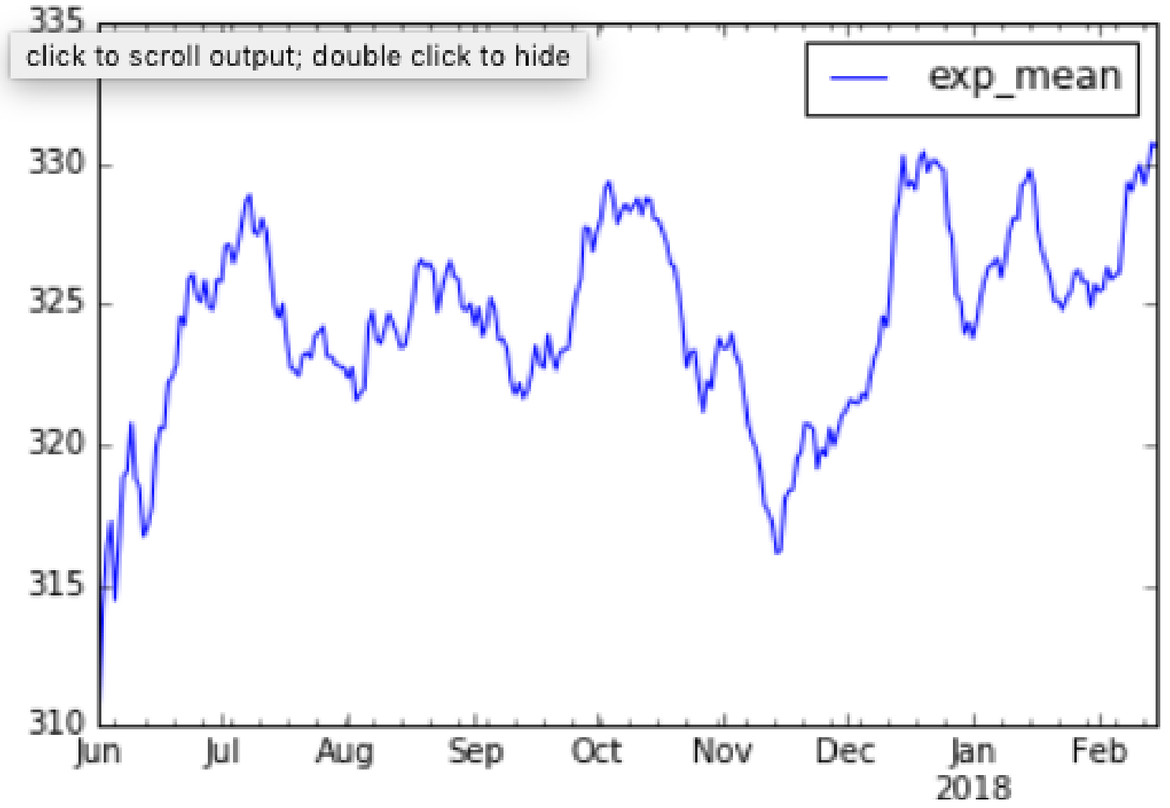


Calculating an Exponential Moving Average

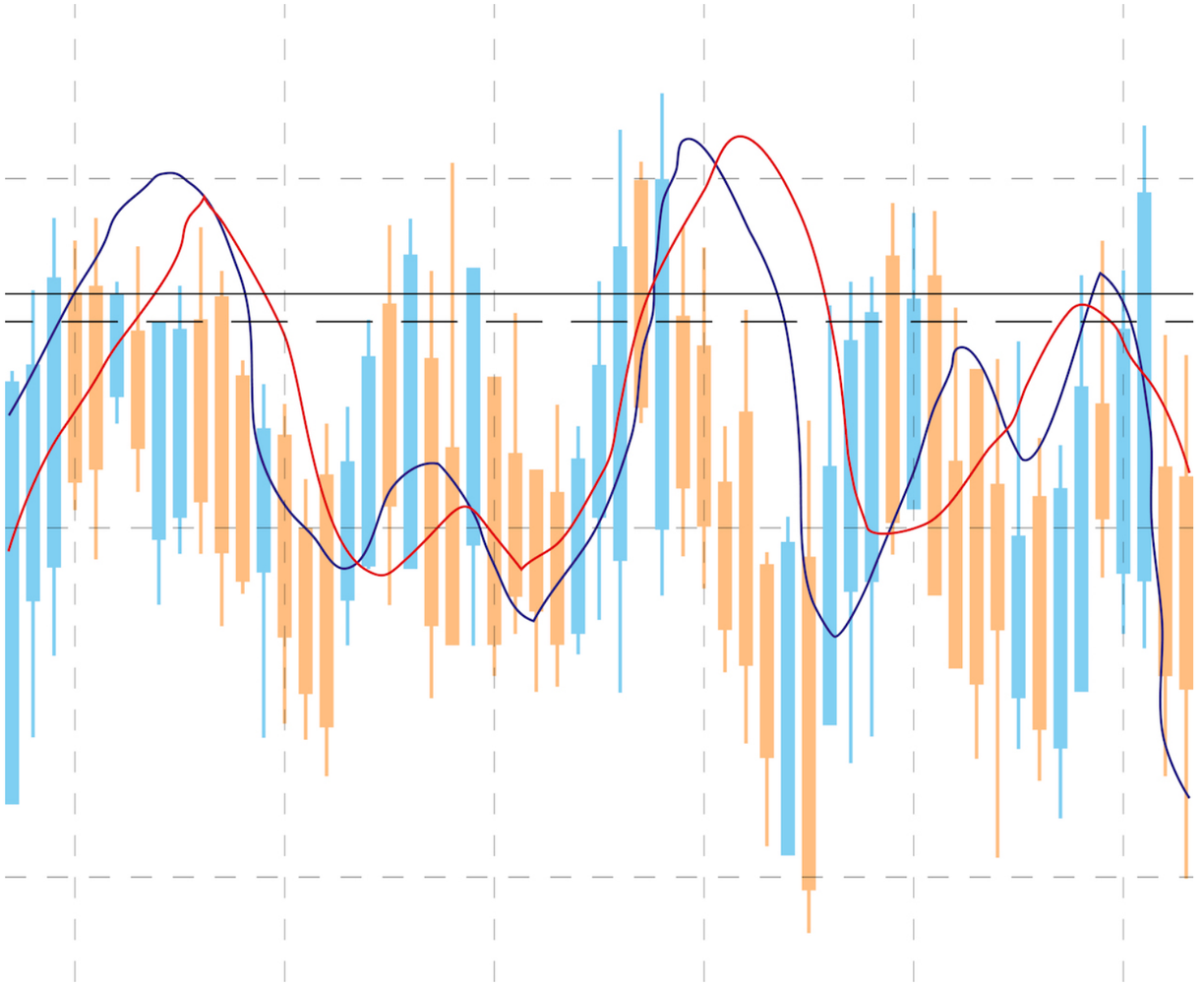
```
exp_mean = high_sku_purchases.purchases.ewm(span=30)

exp_mean = exp_mean.mean()

high_sku_purchases['exp_mean'] = exp_mean
```

Calculating an Exponential Moving Average

```
high_sku_purchases.plot(x='date', y='exp_mean')
```



Data Smoothing Techniques



Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Exploratory data analysis with time series data

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON

Ryan Grossman
Data Scientist, EDO

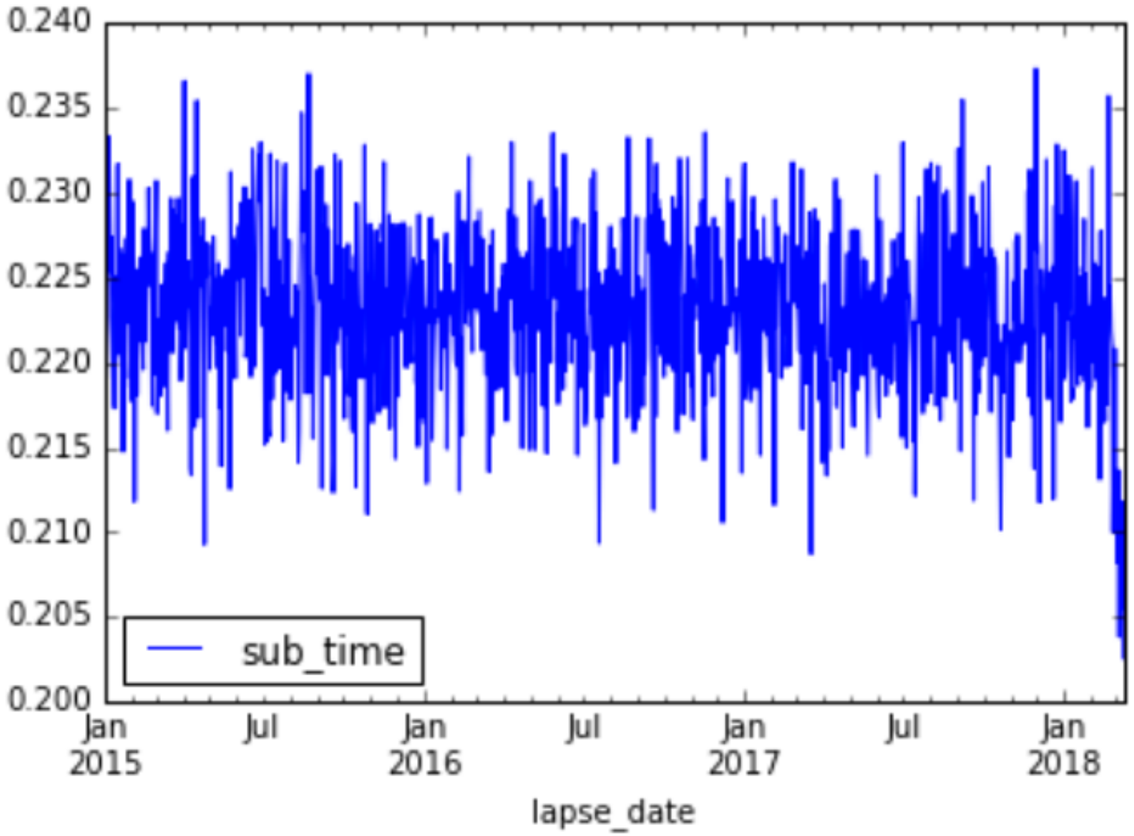
Exploratory Analysis

Drop in New User Retention

```
current_date = pd.to_datetime('2018-03-17')
max_lapse_date = current_date - timedelta(days=7)
conv_sub_data = sub_data_demo[sub_data_demo.lapse_date
                              <= max_lapse_date]
sub_time = (conv_sub_data.subscription_date -
                           conv_sub_data.lapse_date).dt.days
conv_sub_data['sub_time'] = sub_time
conversion_data = conv_sub_data.groupby(by=['lapse_date'],
                              as_index=False)
conversion_data = conversion_data.agg({'sub_time': [gc7]})
conversion_data.columns = conversion_data.columns.droplevel(level=1)
```

```
conversion_data.plot()
plt.show()
```

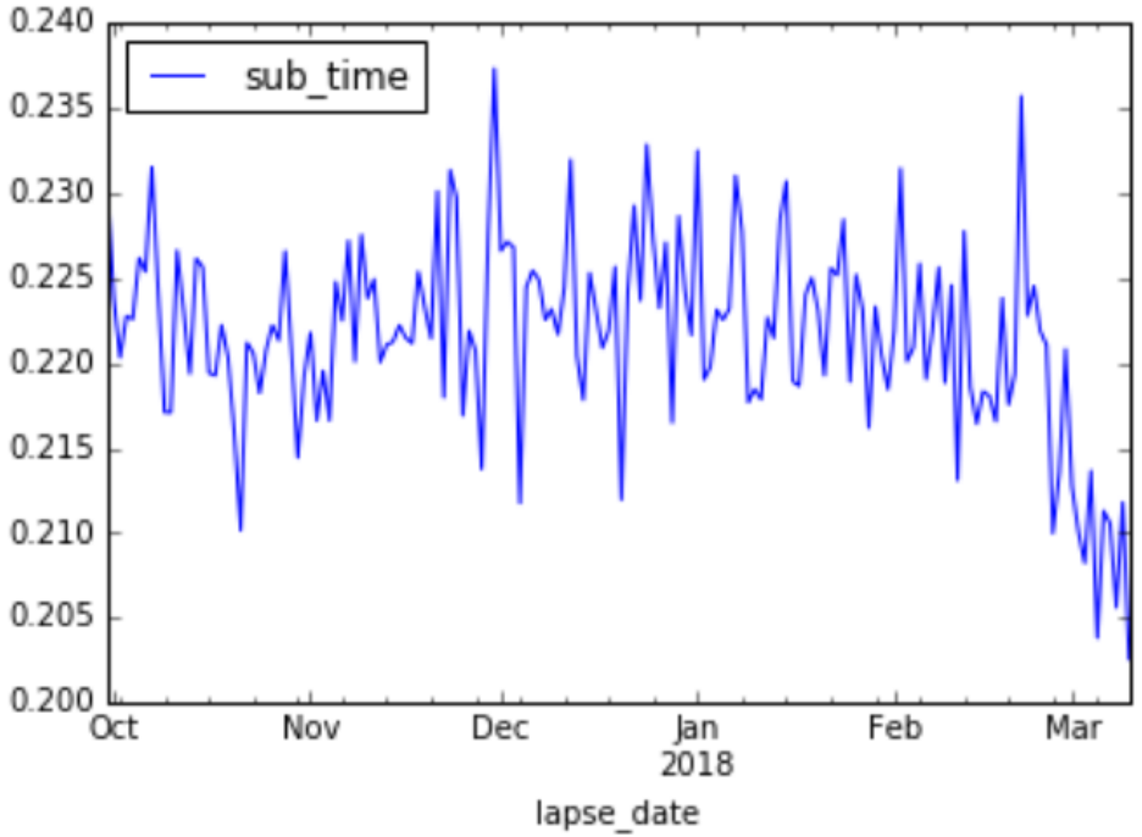
Limiting our View



Limiting our View

```
current_date = pd.to_datetime('2018-03-17')

start_date = current_date - timedelta(days=(6*28))

conv_filter = ((conversion_data.lapse_date >= start_date) &
                           (conversion_data.lapse_date <= current_date))

conversion_data_filt = conversion_data[conv_filter]

conversion_data_filt.plot(x='lapse_date', y='sub_time')

plt.show()
```

Uncovering the Dip



Segmenting our Graph

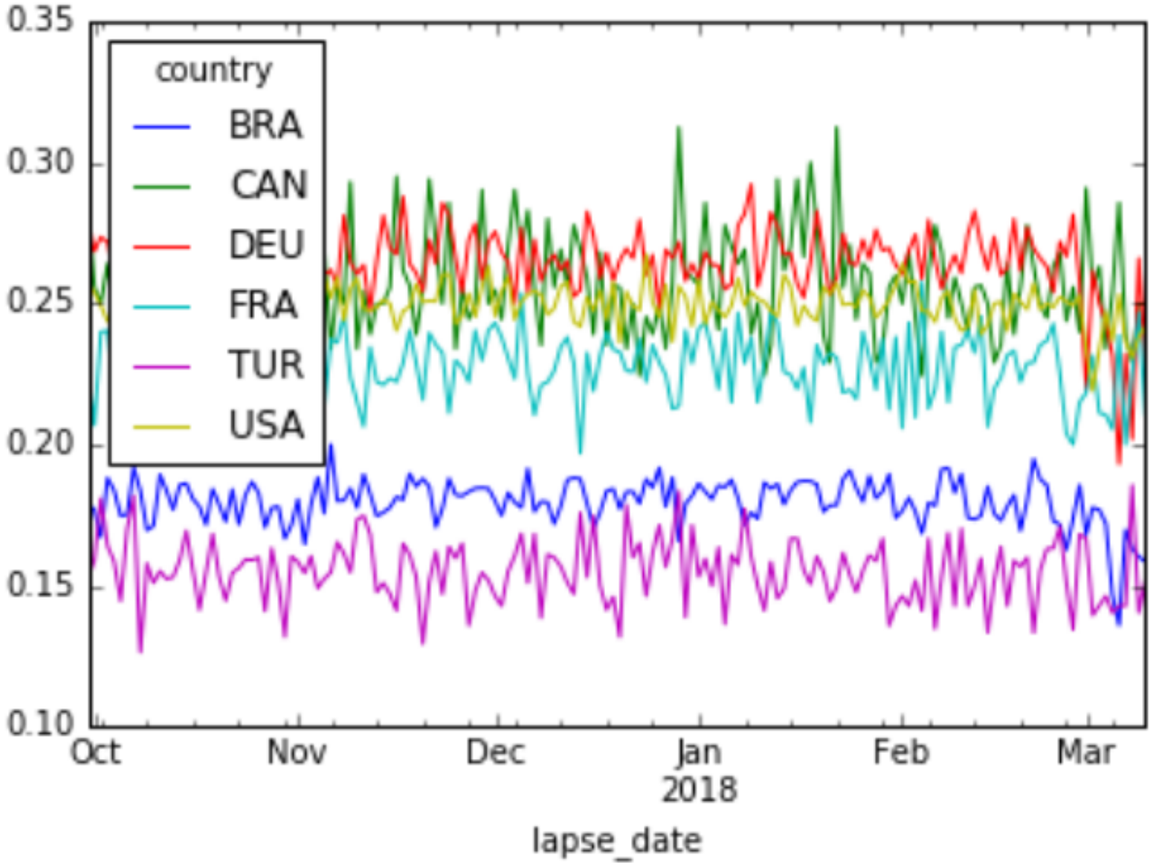
Splitting by Country & Device

```
conv_filter = ((conv_sub_data.lapse_date >= start_date) &
                           (conv_sub_data.lapse_date <= current_date))
conv_data = conv_sub_data[conv_filter]
conv_data_cntry = conv_data.groupby(by=['lapse_date', 'country'],
                                as_index=False)
conv_data_cntry = conv_data_cntry.agg({'sub_time': [gc7]})
conv_data_cntry.columns = conv_data_cntry.columns.droplevel(level=1)
conv_data_cntry = pd.pivot_table(conv_data_cntry,
                                values=['sub_time'],
                                columns=['country'],
                                index=['lapse_date'],fill_value=0 )
conv_data_cntry.columns = conv_data_cntry.columns.droplevel(
                                    level=0)
conv_data_cntry.reset_index(inplace=True)
conv_data_cntry.plot(x=['lapse_date'],
            y=['BRA', 'CAN', 'DEU', 'FRA', 'TUR', 'USA'])
plt.show()
```

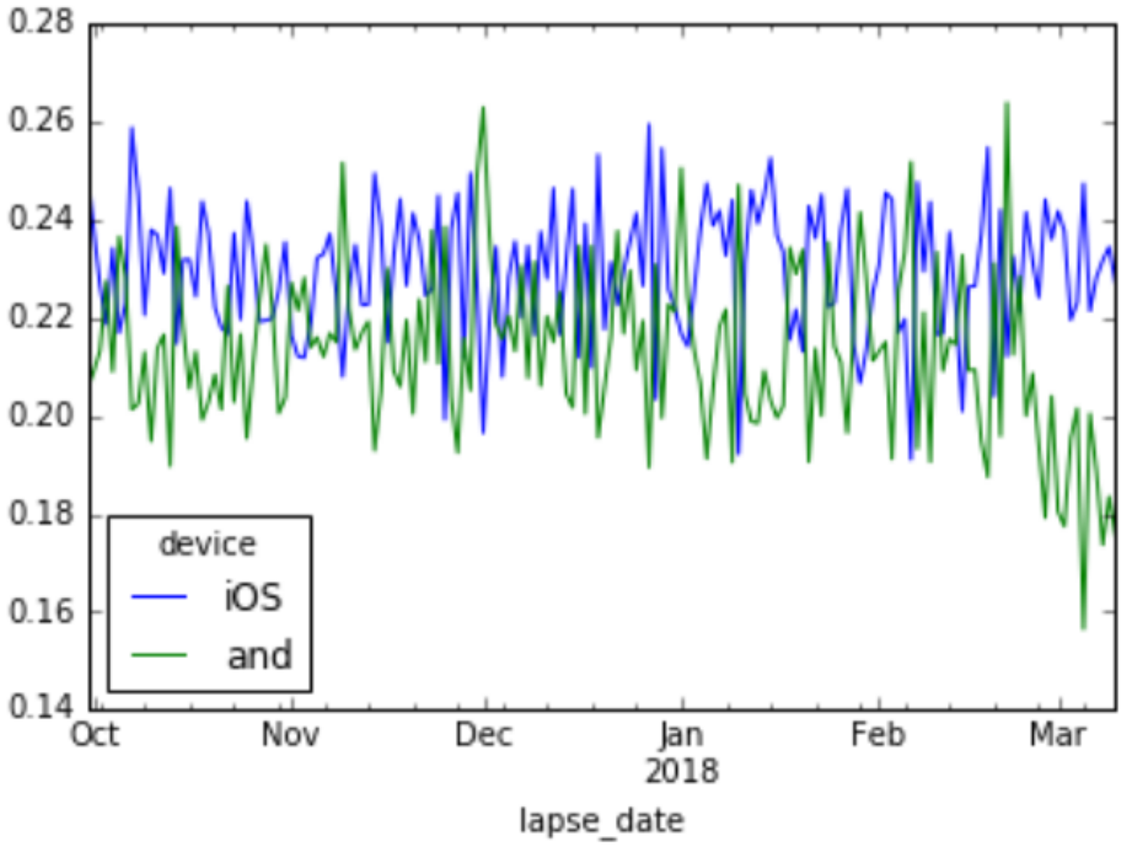
Splitting by Country & Device

```
conv_filter = ((conv_sub_data.lapse_date >= start_date) &
                           (conv_sub_data.lapse_date <= current_date))
conv_data = conv_sub_data[conv_filter]
conv_data_dev = conv_data.groupby(by=['lapse_date',
                           'device'], as_index=False)
conv_data_dev = conv_data_dev.agg({'sub_time': [gc7]})
conv_data_dev.columns = conv_data_dev.columns.droplevel(level=1)
conv_data_dev = pd.pivot_table(conv_data_dev,
                           values=['sub_time'],
                           columns=['device'],
                           index=['lapse_date'],fill_value=0 )
conv_data_dev.columns = conv_data_dev.columns.droplevel(level=[0])
conv_data_dev.reset_index(inplace=True)
conv_data_dev.plot(x=['lapse_date'],
                   y=['iOS', 'and'])
plt.show()
```

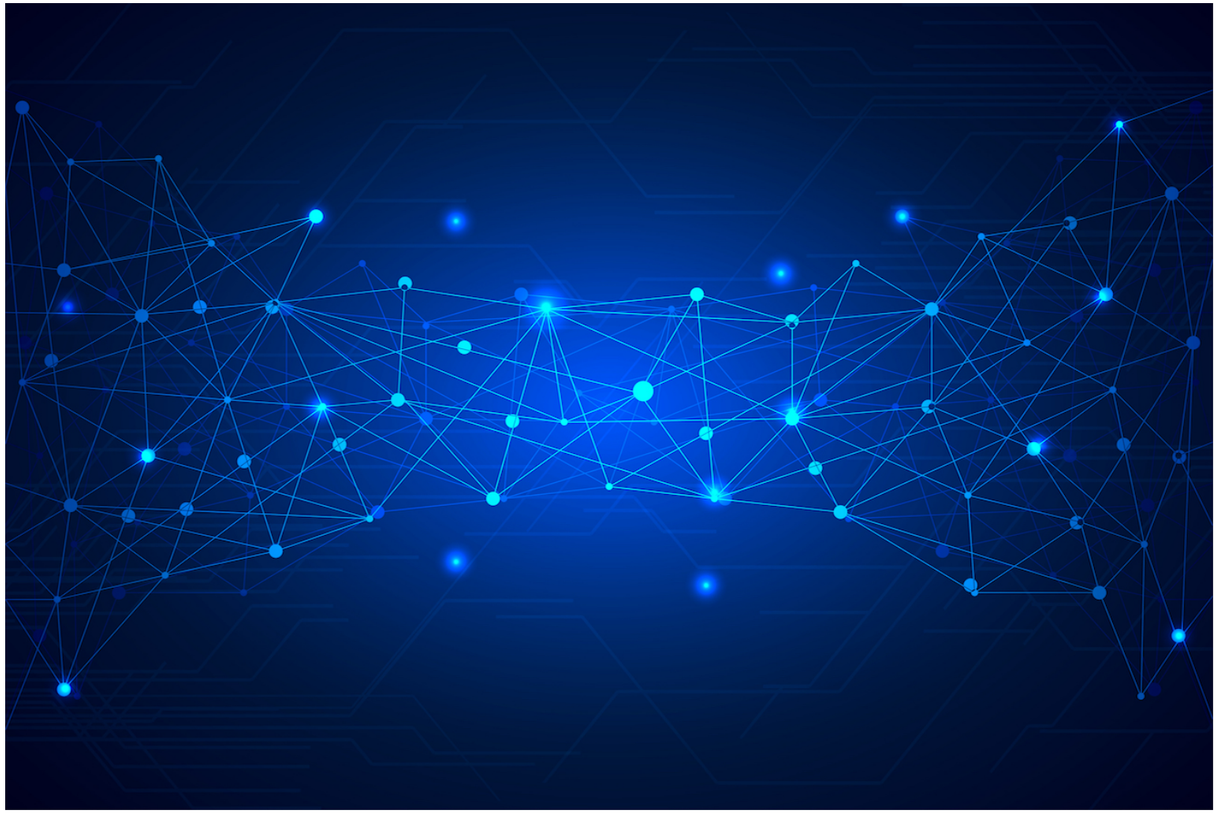
Breaking out by Country



Breaking Out by Device



Adding Annotations



Annotation Datasets

```
events = pd.read_csv('events.csv')
events.head()
```

```
Date            Event
2018-01-01        NYD
2017-01-01        NYD
2016-01-01        NYD
2015-01-01        NYD
2014-01-01        NYD
```

```
releases = pd.read_csv('releases.csv')
releases.head()
```

```
    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
```

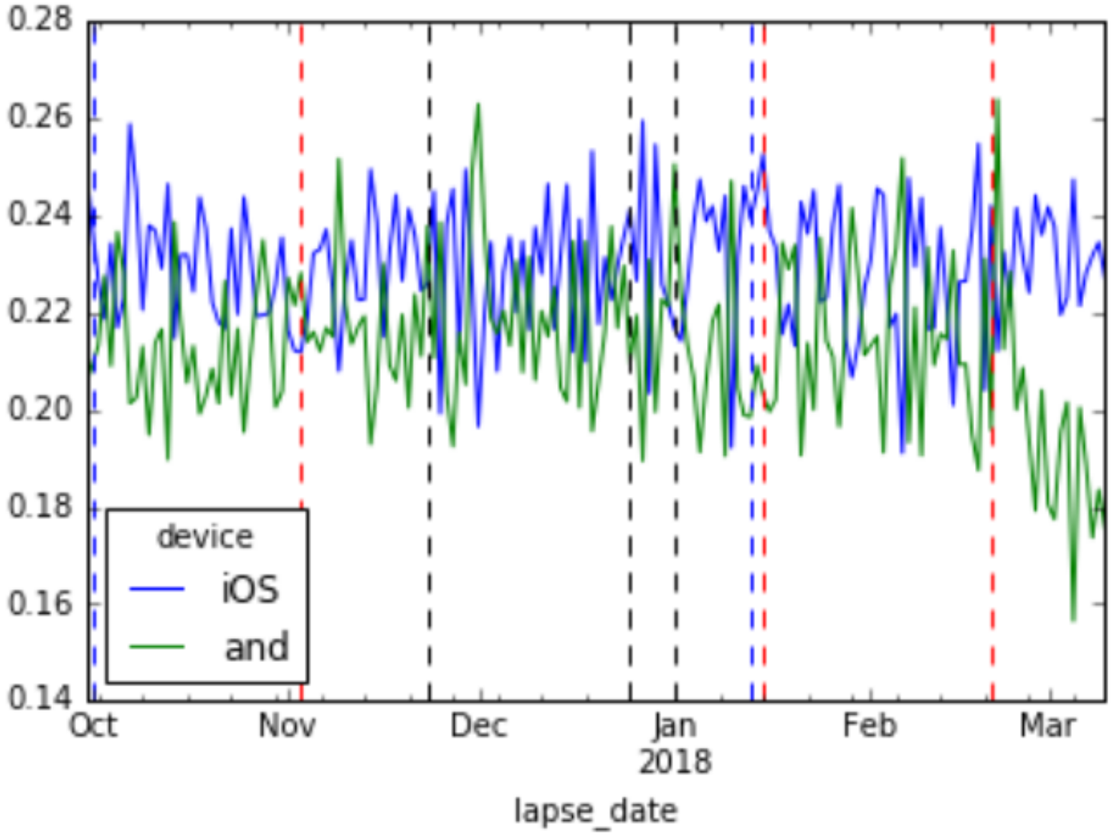
Plotting Annotations

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

events.Date = pd.to_datetime(events.Date)

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

Plotting Annotations

```
releases.Date = pd.to_datetime(releases.Date)

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='--')

plt.show()
```

Our Final Plot



Exploratory Analysis

Let's practice!

CUSTOMER ANALYTICS & A/B TESTING IN PYTHON