ANALYZING SOCIAL MEDIA DATA IN PYTHON

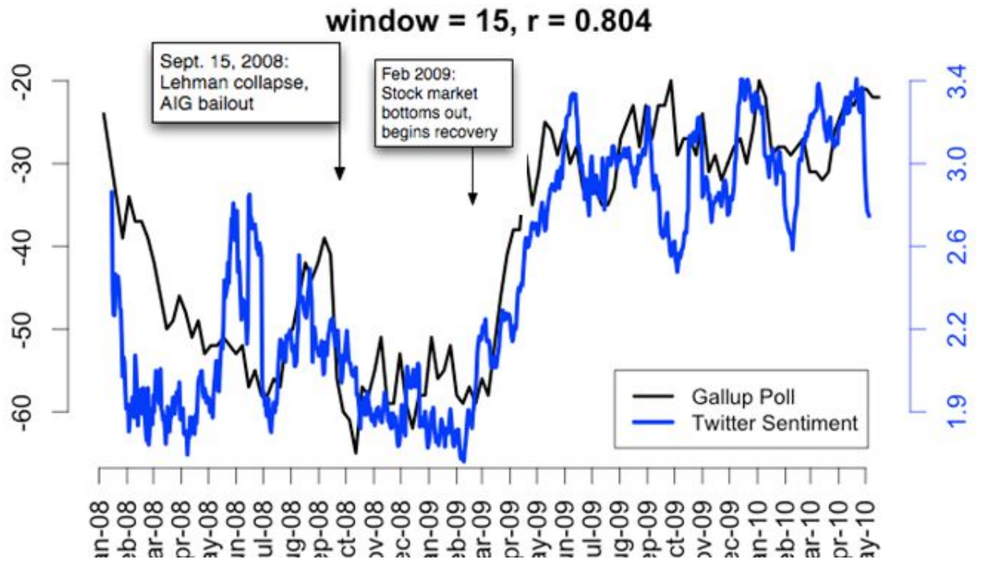
Analyzing Twitter Data

Alex Hanna
Computational Social Scientist

Why Analyze Twitter Data?

Twitter sentiment versus Gallup Poll of Consumer Confidence

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010.
From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In ICWSM-2010



Why Analyze Twitter Data?

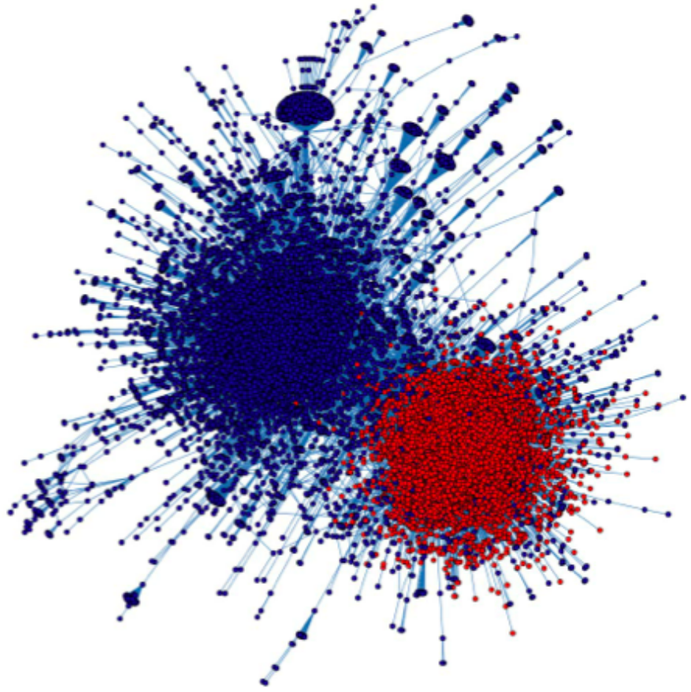


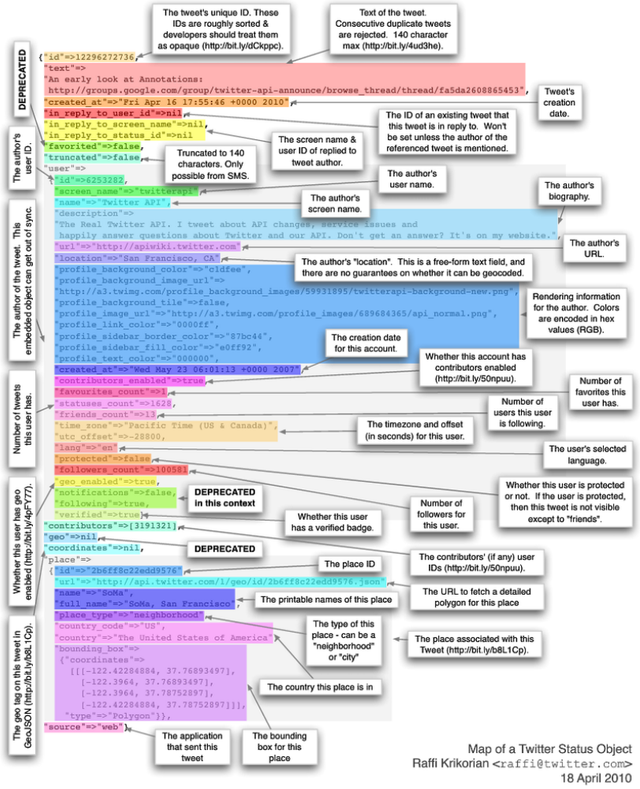
Fig. 2. The political retweet network, laid out using a force-directed algorithm. Node colors reflect cluster assignments (see text).

Source: Conover et al. (2011)

What you can't analyze

- Can't collect data on observers
- Free-level of access is restrictive
  - Can't collect historical data
  - Only a 1% (unverified) sample

What you can analyze



- 1% sample is still a few million tweets
- Within a tweet
  - Text
  - User profile information
  - Geolocation
  - Retweets and quoted tweets

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's review!

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Collecting data through the Twitter API

Alex Hanna
Computational Social Scientist

Twitter API

- API: Application Programming Interace
  - Method of accessing data
- Twitter APIs
  - Search API
  - Ads API
  - Streaming API

Streaming API

- Streaming API
  - Real-time tweets
- Filter endpoint
  - Keywords
  - User IDs
  - Locations
- Sample endpoint
  - Random sample

Using tweepy to collect data

- `tweepy`
  - Python package for accessing Streaming API

SListener

```
from tweepy.streaming import StreamListener
import time

class SListener(StreamListener):
    def __init__(self, api = None):
        self.output  = open('tweets_%s.json' %
            time.strftime('%Y%m%d-%H%M%S'), 'w')
        self.api = api or API()
    ...
```

tweepy authentication

```
from tweepy import OAuthHandler
from tweepy import API

auth = OAuthHandler(consumer_key, consumer_secret)

auth.set_access_token(access_token, access_token_secret)

api = API(auth)
```

Collecting data with tweepy

```
from tweepy import Stream

listen = SListener(api)

stream = Stream(auth, listen)

stream.sample()
```

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's practice!

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Understanding Twitter JSON

Alex Hanna
Computational Social Scientist

Contents of Twitter JSON

```
{
        "created_at": "Thu Apr 19 14:25:04 +0000 2018",
        "id": 986973961295720449,
        "id_str": "986973961295720449",
        "text": "Writing out the script of my @DataCamp class
            and I can't help but mentally read it back to myself in
            @hugobowne's voice.",
        "retweet_count": 0,
        "favorite_count": 1,
        ...
}
```

- How many retweets, favorites
- Language
- Reply to which tweet
- Reply to which user

Child JSON objects

```
{
        "user": {
            "id": 661613,
            "name": "Alex Hanna, Data Witch",
            "screen_name": "alexhanna",
            "location": "Toronto, ON",
            ...
         }
}
```

Places, retweets/quoted tweets, and 140+ tweets

- `place` and `coordinate`
  - contain geolocation
- `extended_tweet`
  - tweets over 140 characters
- `retweeted_status` and `quoted_status`
  - contain all tweet information of retweets and quoted tweets

Accessing JSON

```
import json

tweet_json = open('tweet-example.json', 'r').read()

tweet = json.loads(tweet_json)

tweet['text']
```

Child tweet JSON

```
tweet['user']['screen_name']

tweet['user']['name']

tweet['user']['created_at']
```

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's practice!

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Processing Twitter Text

Alex Hanna
Computational Social Scientist

Text in Twitter JSON

```
tweet_json = open('tweet-example.json', 'r').read()
tweet = json.loads(tweet_json)
tweet['text']
```

More than 140 characters

```
tweet['extended_tweet']['full_text']
```

Retweets and quoted tweets

```
tweet['quoted_status']['extended_tweet']['full_text']
```

Textual user information

```
tweet['user']['description']
tweet['user']['location']
```

Flattening Twitter JSON

```
extended_tweet['extended_tweet-full_text'] =
    extended_tweet['extended_tweet']['full_text']
```

Flattening Twitter JSON

```
tweet_list = []
with open('all_tweets.json', 'r') as fh:
    tweets_json = fh.read().split("\n")

    for tweet in tweets_json:
        tweet_obj = json.loads(tweet)

        if 'extended_tweet' in tweet_obj:
            tweet_obj['extended_tweet-full_text'] =
                tweet_obj['extended_tweet']['full_text']
        ...

    tweet_list.append(tweet)

tweets = pd.DataFrame(tweet_list)
```

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's practice!

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Counting words

Alex Hanna
Computational Social Scientist

Why count words?

- Basic step for automation of text analysis
- Can tell us how many times a relevant keyword is mentioned in documents in comparison to others
- In exercises: `#rstats` vs `#python`

Counting with str.contains

- `str.contains`
  - pandas `Series` string method
  - Returns boolean `Series`
  - `case = False` - Case insensitive search

Companies dataset

```
> import pandas as pd

> tweets = pd.DataFrame(flatten_tweets(companies_json))

> apple = tweets['text'].str.contains('apple', case = False)

> print(np.sum(apple) / tweets.shape[0])

0.112
```

Counting in multiple text fields

```
> apple = tweets['text'].str.contains('apple', case = False)

> for column in ['extended_tweet-full_text',
    'retweeted_status-text',
    'retweeted_status-extended_tweet-full_text']:
    apple = apple | tweets[column].str.contains('apple', case = False)

> print(np.sum(apple) / tweets.shape[0])

0.12866666666666668
```

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's practice!

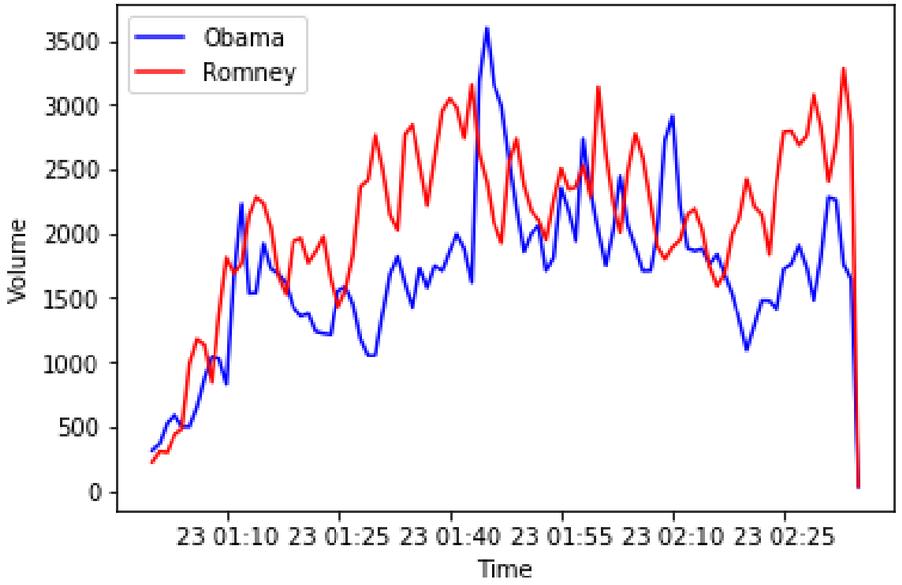
ANALYZING SOCIAL MEDIA DATA IN PYTHON

Time Series

Alex Hanna
Computational Social Scientist

Time series data

```
                     sum person
date
2012-10-23 01:00:00  314  Obama
2012-10-23 01:01:00  369  Obama
2012-10-23 01:02:00  527  Obama
2012-10-23 01:03:00  589  Obama
2012-10-23 01:04:00  501  Obama
...
```



Converting datetimes

```
> print(tweets['created_at'])

0       Sat Jan 27 18:36:21 +0000 2018
1       Sat Jan 27 18:24:02 +0000 2018
2       Sat Jan 27 18:09:14 +0000 2018
...

> tweets['created_at'] = pd.to_datetime(tweets['created_at'])

> print(tweets['created_at'])

0      2018-01-27 18:36:21
1      2018-01-27 18:24:02
2      2018-01-27 18:09:14
...

> tweets = tweets.set_index('created_at')
```

Keywords as time series metrics

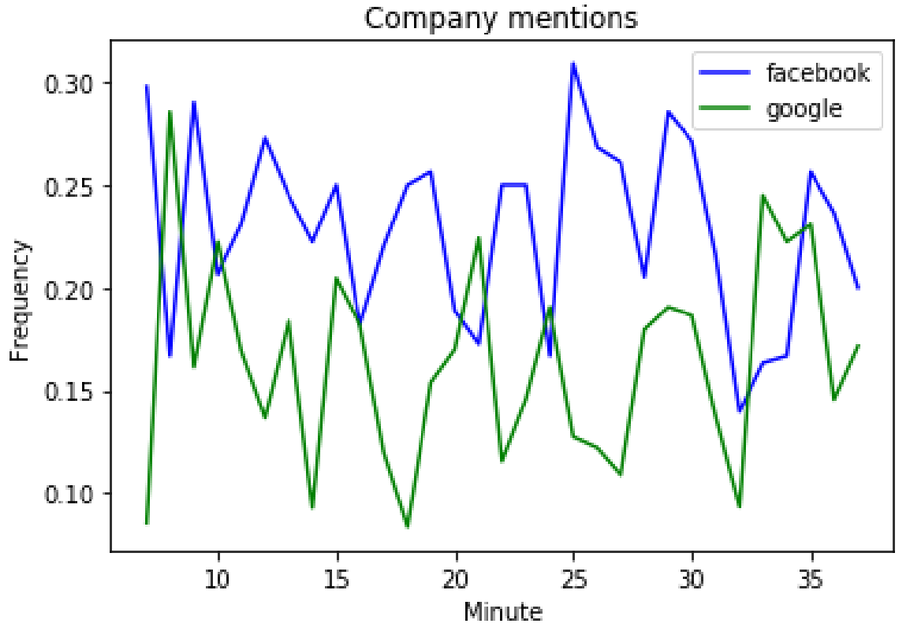
```
> tweets['google'] = check_word_in_tweet('google', tweets)

> print(tweets['google'])

created_at
2018-01-27 18:36:21    False
2018-01-27 18:24:02    False
2018-01-27 18:30:12    False
2018-01-27 18:12:37     True
2018-01-27 18:11:06     True
....

> print(np.sum(tweets['google']))

247
```

Generating keyword means

```
> mean_google = tweets['google'].resample('1 min').mean()

> print(mean_google)

created_at
2018-01-27 18:07:00    0.085106
2018-01-27 18:08:00    0.285714
2018-01-27 18:09:00    0.161290
2018-01-27 18:10:00    0.222222
2018-01-27 18:11:00    0.169231
```

Plotting keyword means

```
import matplotlib.pyplot as plt

plt.plot(means_facebook.index.minute,
    means_facebook, color = 'blue')
plt.plot(means_google.index.minute,
    means_google, color = 'green')
plt.xlabel('Minute')
plt.ylabel('Frequency')
plt.title('Company mentions')
plt.legend(('facebook', 'google'))
plt.show()
```



ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's practice!

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Sentiment Analysis

Alex Hanna
Computational Social Scientist

Understanding sentiment analysis

- Method
  - Counting positive/negative words in the document
  - Assessing positivity/negativity of the whole document
- Uses
  - Analyzing reactions to a company, product, politician, or policy

Sentiment analysis tools

- VADER `SentimentIntensityAnalyzer()`
  - Part of Natural Language Toolkit (`nltk`)
  - Good for short texts like tweets
  - Measures sentiment of particular words (e.g. angry, happy)
  - Also considers sentiment of emoji (☐☐) and capitalization (Nice vs NICE)

Implementing sentiment analysis

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()

sentiment_scores = tweets['text'].apply(sid.polarity_scores)
```

Interpreting sentiment scores

- Reading tweets as part of the process
  - Does it have *face validity*? (i.e. does this match my idea of what it means to be positive or negative?)

Interpreting sentiment scores

```
tweet1 = 'RT @jeffrey_heer: Thanks for inviting me, and thanks for the
lovely visualization of the talk! ...'
print(sid.polarity_scores(tweet1))

{'neg': 0.0, 'neu': 0.496, 'pos': 0.504, 'compound': 0.9041}
```

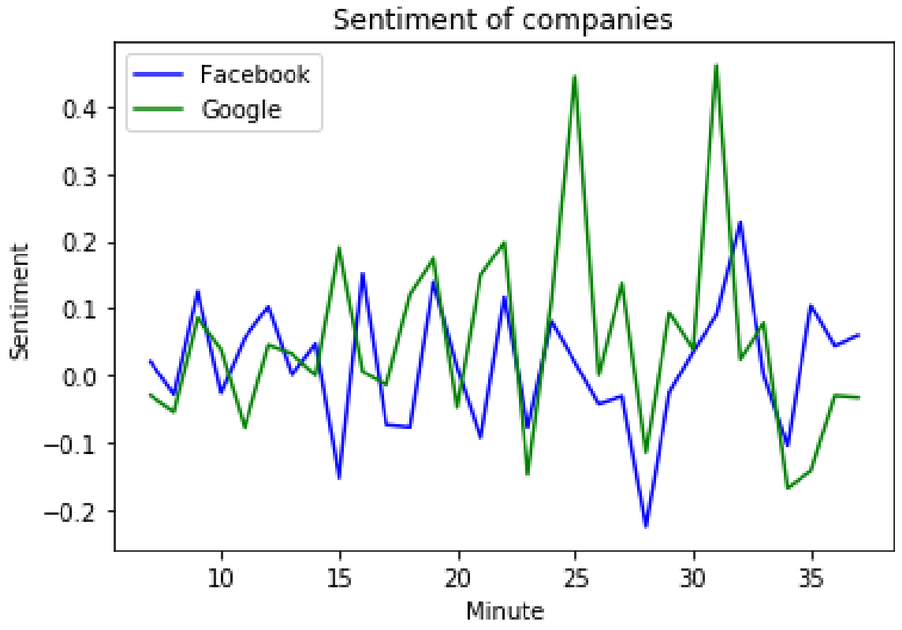
```
tweet2 = 'i am having problems with google play music'
print(sid.polarity_scores(tweet2)

{'neg': 0.267, 'neu': 0.495, 'pos': 0.238, 'compound': -0.0772}
```

Generating sentiment averages

```
sentiment = sentiment_scores.apply(lambda x: x['compound'])

sentiment_fb = sentiment[check_word_in_tweet('facebook', tweets)]
                    .resample('1 min').mean()
sentiment_gg = sentiment[check_word_in_tweet('google', tweets)]
                    .resample('1 min').mean()
```

Plotting sentiment scores

```
plt.plot(sentiment_fb.index.minute,
         sentiment_fb, color = 'blue')
plt.plot(sentiment_g.index.minute,
         sentiment_gg, color = 'green')

plt.xlabel('Minute')
plt.ylabel('Sentiment')
plt.title('Sentiment of companies')
plt.legend(('Facebook', 'Google'))
plt.show()
```



ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's practice!

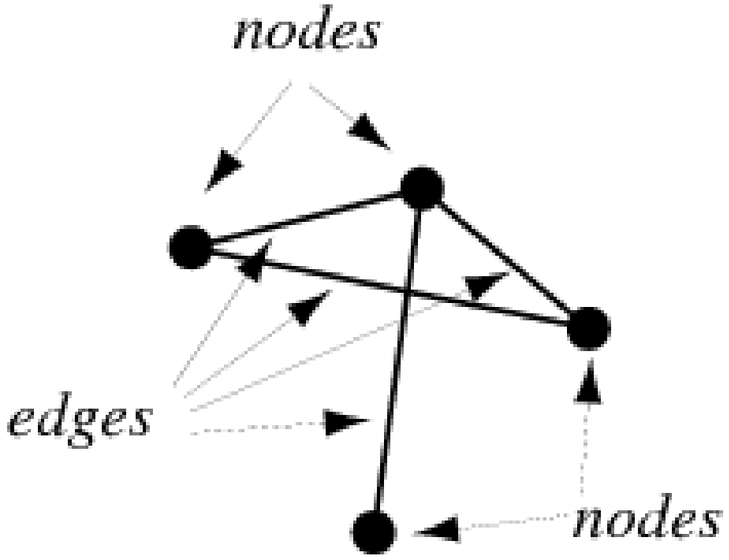
ANALYZING SOCIAL MEDIA DATA IN PYTHON

Twitter Networks

Alex Hanna
Computational Social Scientist


14
Source: AP

Network analysis: terms



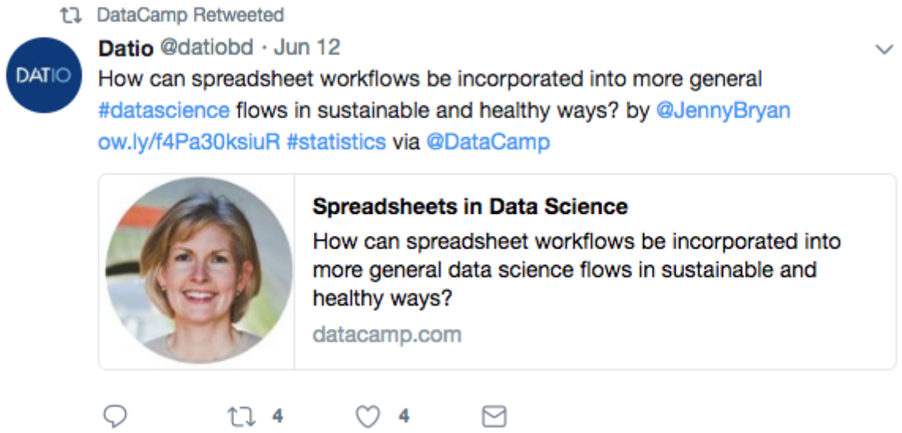
- Directed networks
  - Relationships are not mutual
- Source node
  - Where the arrow starts
- Target node
  - Where the arrow edges

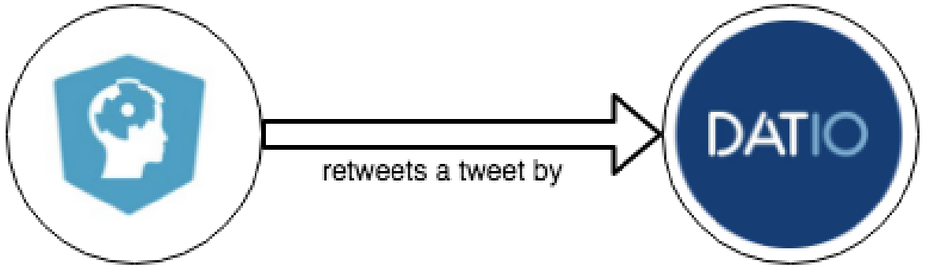
Source:
http://mathworld.wolfram.com/GraphEdge.html

Types of Twitter network ties

- Twitter networks
  - Retweets
  - Quotes
  - Replies

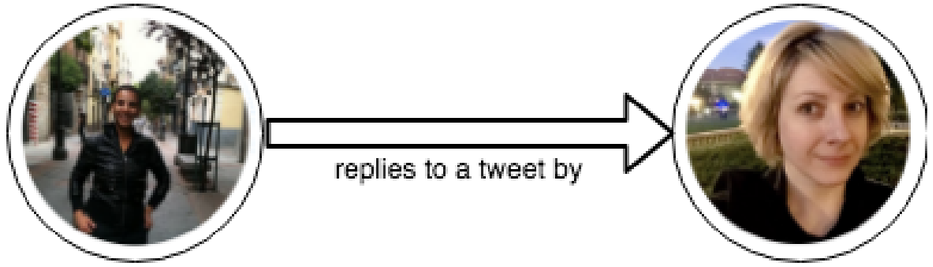
Retweet networks





Quote networks

Reply networks



ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's practice!

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Importing and visualizing Twitter networks

Alex Hanna
Computational Social Scientist

Edge Lists

```
         BethMohn          ChristianMohn
           ASilNY       LarrySchweikart
         mattg444           WhiteHouse
   hlthiskrieger             aravosis
        Herky86        SenJeffMerkley
PatrickParsons9            TwitterGov
  New_Narrative               CFR_org
         dddlor            roywoodjr
     scrivener50        michaelscherer
ChiefsHeadCoach          johnpavlovitz
```

Importing a retweet network

```
import networkx as nx

## ... flatten and convert JSON

G_rt = nx.from_pandas_edgelist(
                          tweets,
                          source = 'user-screen_name',
                          target = 'retweeted_status-user-screen_name',
                          create_using = nx.DiGraph())
```

Importing a quoted network

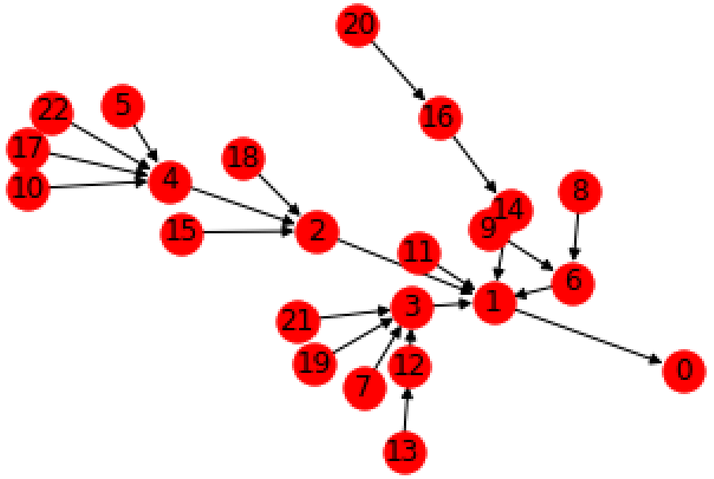
```
import networkx as nx

## ... flatten and convert JSON

G_quote = nx.from_pandas_edgelist(
                            tweets,
                            source = 'user-screen_name',
                            target = 'quoted_status-user-screen_name',
                            create_using = nx.DiGraph())
```

Importing a reply network

```
import networkx as nx

## ... flatten and convert JSON

G_reply = nx.from_pandas_edgelist(
                            tweets,
                            source = 'user-screen_name',
                            target = 'in_reply_to_screen_name'
                            create_using = nx.DiGraph())
```

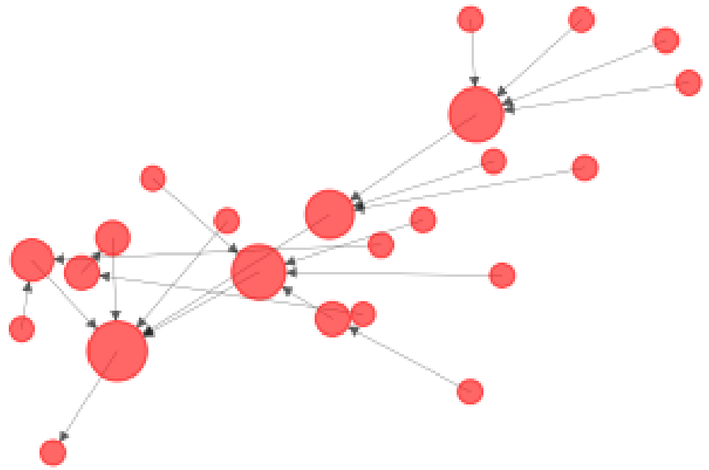
Visualization

```
nx.draw_networkx(T)
plt.axis('off')
```



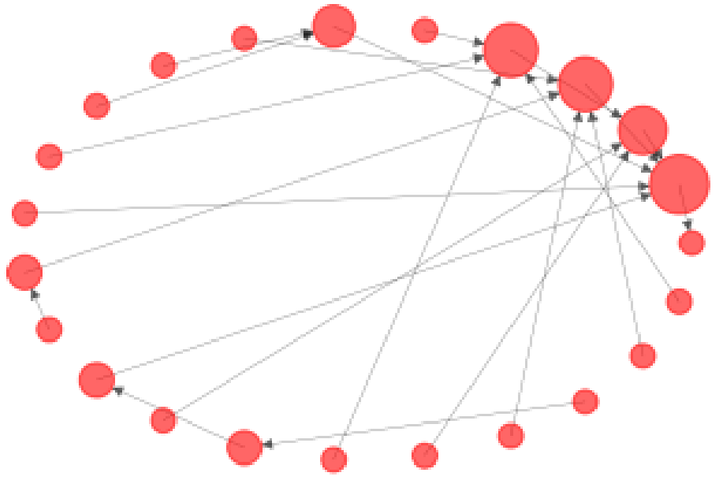
Visualization options

```
sizes =
    [x[1]*100 for x in T.degree()]
nx.draw_networkx(T,
                 node_size = sizes,
                 with_labels = False,
                 alpha = 0.6,
                 width = 0.3)
plt.axis('off')
```



Circular layout

```
circle_pos =
nx.circular_layout(T)
nx.draw_networkx(T,
                 pos = circle_pos,
                 node_size = sizes,
                 with_labels = False,
                 alpha = 0.6,
                 width = 0.3)
plt.axis('off')
```



ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's practice!

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Node-level metrics

Alex Hanna
Computational Social Scientist

Centrality: node importance

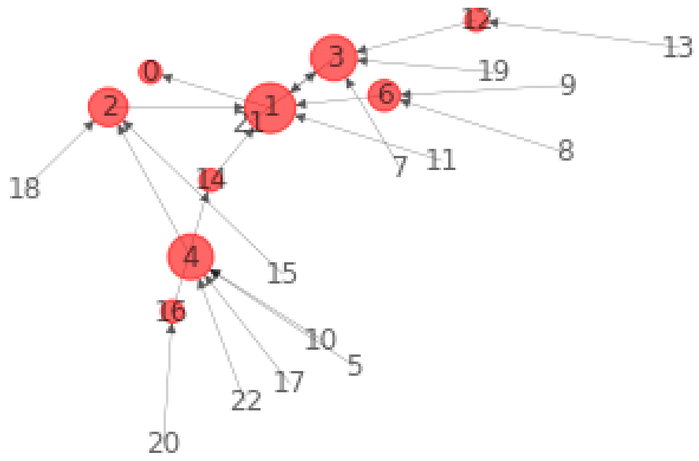
- Centrality
  - Measures of importance of a node in a network
  - Several different ideas of "importance"

Degree Centrality

Degree

```
nx.in_degree_centrality(T)
nx.out_degree_centrality(T)
```

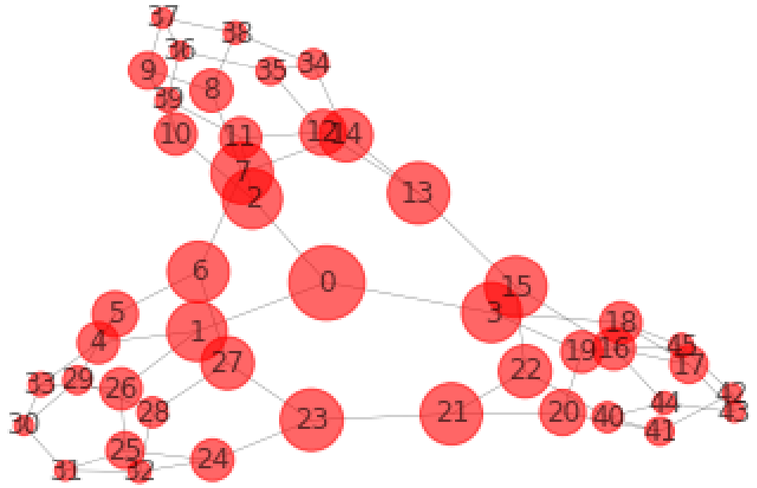
- Number of edges that are connected to node
- Two types of degrees in a directed network
  - In-degree - edge going **into** node
  - Out-degree - edge going **out of** a node



Betweenness Centrality

- How many shortest paths between two nodes pass through this node
- Importance as a network broker

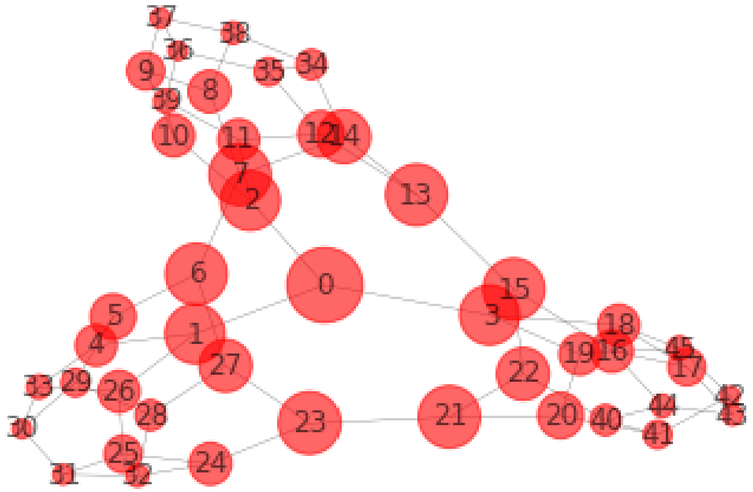
```
nx.betweenness_centrality(T)
```



Printing highest centrality

```
bc = nx.betweenness_centrality(T)

betweenness = pd.DataFrame(
          list(bc.items()),
          columns = ['Name', 'Cent'])

print(betweenness.sort_values(
          'Cent',
          ascending = False).head())
```

```
    Name  Centrality
0      0    0.232540
23    23    0.158514
7      7    0.158514
15    15    0.158514
21    21    0.157588
```



Centrality in different networks

		Centrality		
		In-Degree	**Out-Degree**	**Betweenness**
Network Type	*Retweets*	Gets retweets	Shares retweets	Bridges different topic/ideology communities
	Replies	Gets most replies	Participates in many conversations	Bridges different topic/ideology discussions

The Ratio

```
degree_rt = pd.DataFrame(list(G_rt.in_degree()),
                         columns = ['screen_name', 'degree'])
degree_reply = pd.DataFrame(list(G_reply.in_degree()),
                            columns = ['screen_name', 'degree'])

ratio = degree_rt.merge(degree_reply,
                        on = 'screen_name',
                        suffixes = ('_rt', '_reply'))

ratio['ratio'] = ratio['degree_reply'] / ratio['degree_rt']
```

ANALYZING SOCIAL MEDIA DATA IN PYTHON

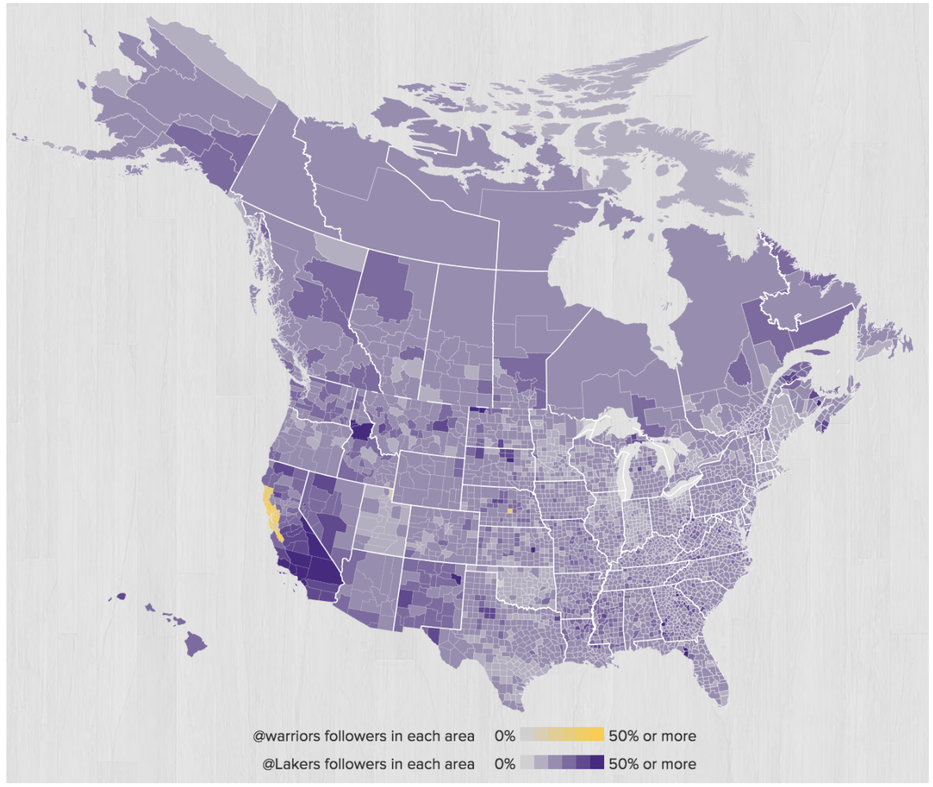
Let's practice!

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Maps and Twitter data

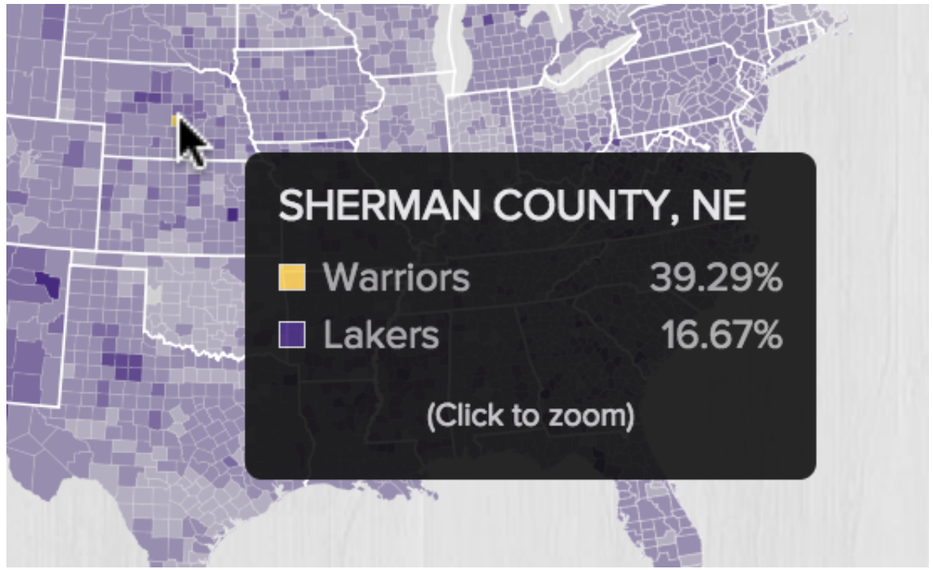
Alex Hanna
Computational Social Scientist

Why maps?



- Geographical scope
  - Participants or observers?
- Differentiating tweets
  - For or against?

How Twitter gets location data



- Location is device-dependent
- In practice, aggregate geographical to county, state-level

Beware selection biases!

- *Warning*: only 1-3% of Twitter data have geographical data
- Limits the generalizability of inference

Types of Geographical Data available in Twitter

- Twitter text (most imprecise)
- User location
- Bounding boxes
- Coordinates and points (most precise)

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's practice!

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Geographical Data in Twitter JSON

Alex Hanna
Computational Social Scientist

Locations in Twitter text

Dr. Alex Hanna, Skatin Data Witch
@alexhanna

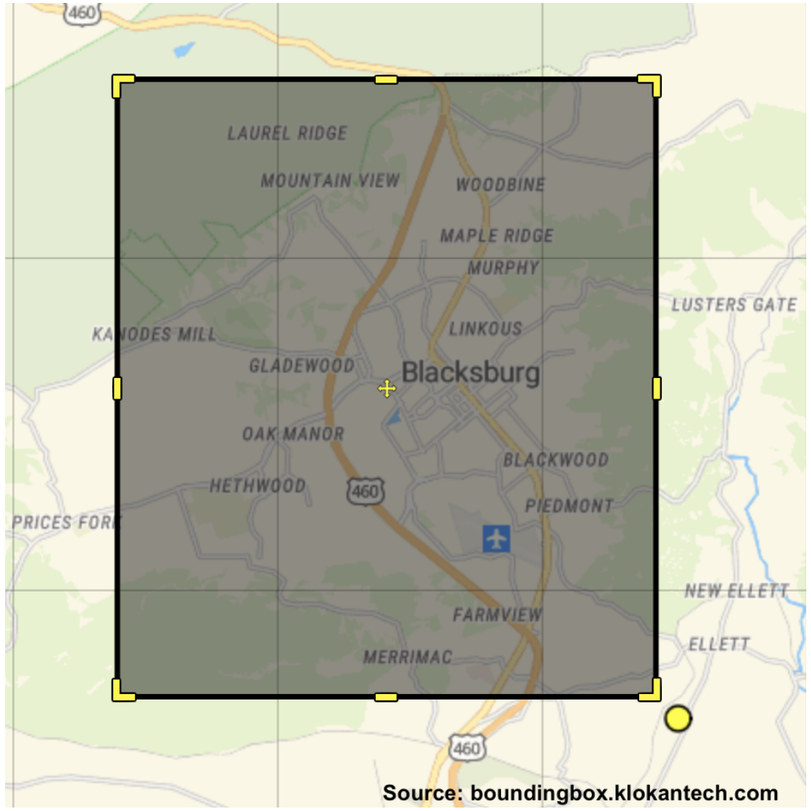
In Zurich! It's lovely and about as hot as Toronto.

12:32 PM - 4 Jul 2018

User-defined location

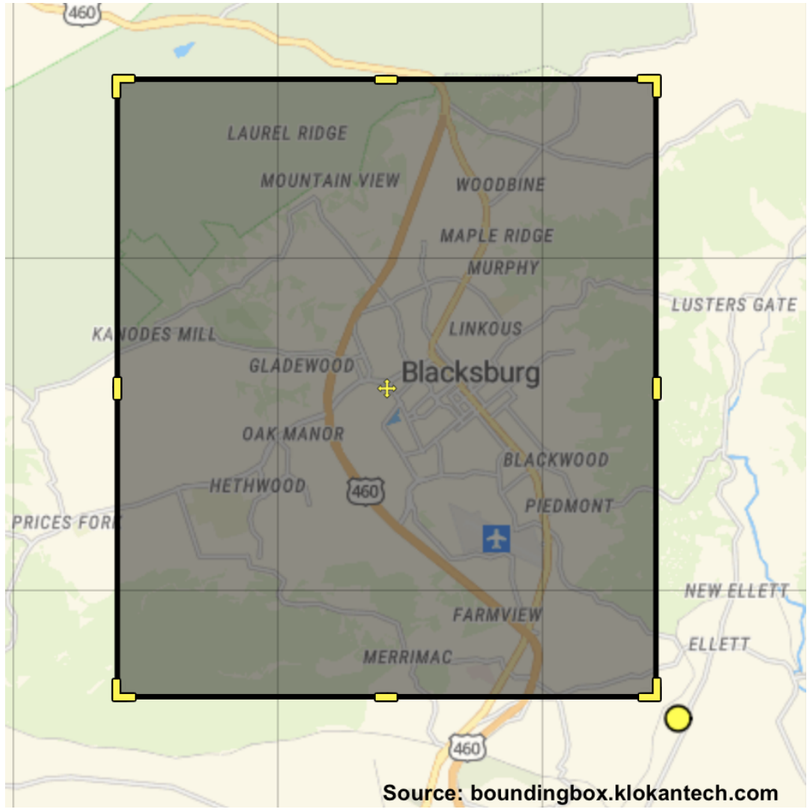

```
> print(tweet['user']['location'])

Bay Area
```

place JSON




```
> print(tweet['place'])

{'attributes': {},
 'bounding_box': {'coordinates':
  [[[-80.47611, 37.185195],
    [-80.47611, 37.273387],
    [-80.381618, 37.273387],
    [-80.381618, 37.185195]]],
  'type': 'Polygon'},
 'country': 'United States',
 'country_code': 'US',
 'full_name': 'Blacksburg, VA',
 'name': 'Blacksburg',
 'place_type': 'city',
 ...}
```

Calculating the centroid




```
coordinates = [
    [-80.47611, 37.185195],
    [-80.47611, 37.273387],
    [-80.381618, 37.273387],
    [-80.381618, 37.185195]]

longs = np.unique( [x[0] for x
    in coordinates] )
lats  = np.unique( [x[1] for x
    in coordinates] )

central_long = np.sum(longs) / 2
central_lat  = np.sum(lats) / 2
```

coordinates JSON


```
> print(tweet['coordinates'])

{'type': 'Point',
'coordinates': [-72.2833, 21.7833]}
```

ANALYZING SOCIAL MEDIA DATA IN PYTHON

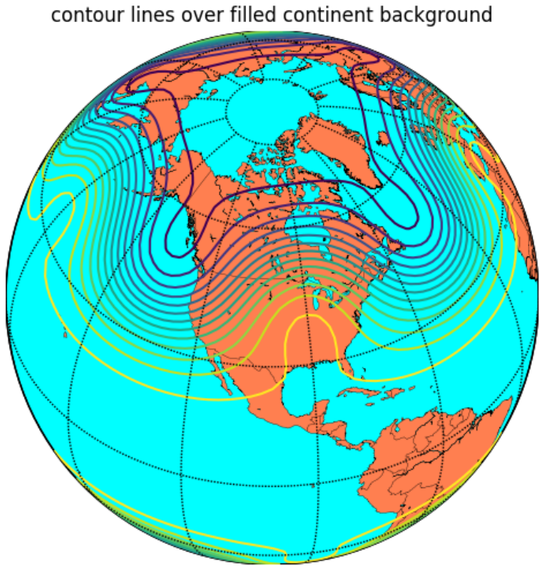
Let's practice!

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Creating Twitter maps

Alex Hanna
Computational Social Scientist

Introducing Basemap



Source: https://matplotlib.org/basemap/users/examples.html

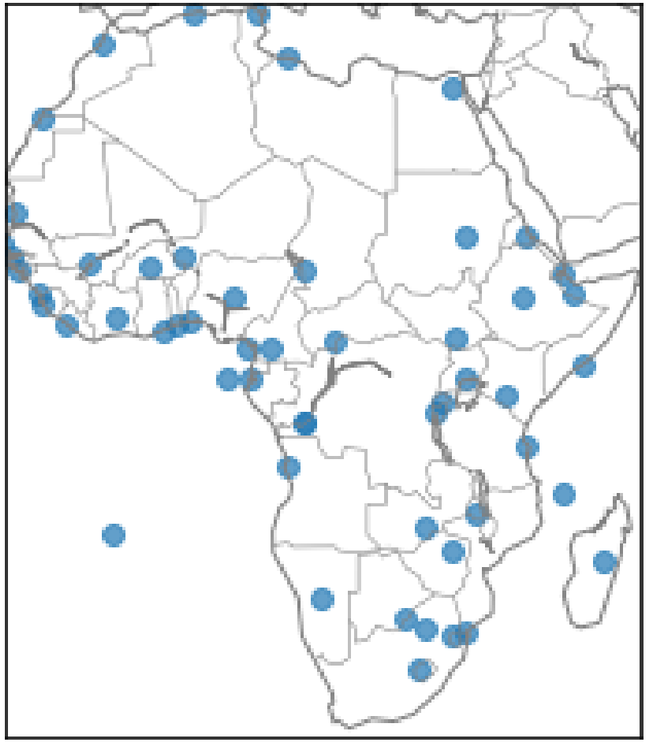
- Library for plotting two-dimensional maps
- Built on top of matplotlib
- Converts coordinates into map projections

Beginning with Basemap

```
from mpl_toolkits.basemap
    import Basemap

m = Basemap(projection='merc',
            llcrnrlat = -35.62,
            llcrnrlon = -17.29,
            urcrnrlat = 37.73,
            urcrnrlon = 51.39)

m.fillcontinents(color='white')
m.drawcoastlines(color='gray')
m.drawcountries(color='gray')
```

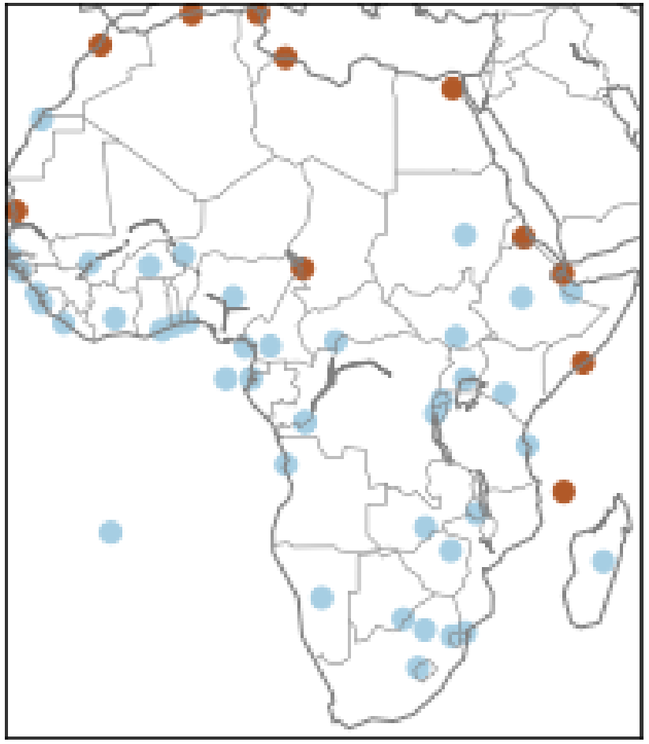
Plotting points

```
africa = pd.read_csv('africa.csv')
longs  = africa['CapitalLongtiude']
lats   = africa['CapitalLatitude']

m = Basemap(...)

m.fillcontinents(color='white',
    zorder = 0)
m.drawcoastlines(color='gray')
m.drawcountries(color='gray')

m.scatter(longs.values,
          lats.values,
          latlon = True,
          alpha = 0.7)
```



Using color

```
africa = pd.read_csv('africa.csv')
longs  = africa['CapitalLongtiude']
lats   = africa['CapitalLatitude']
arabic = africa['Arabic']

m = Basemap(...)
m.fillcontinents(color='white',
    zorder = 0)
m.drawcoastlines(color='gray')
m.drawcountries(color='gray')

m.scatter(longs.values,
          lats.values,
          latlon = True,
          c = arabic.values,
          cmap = 'Paired',
          alpha = 1)
```



ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's practice!

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Congratulations!

Alex Hanna
Computational Social Scientist