In [16]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import csv
import json
import requests
```

In [17]:
```
import sqlalchemy
from sqlalchemy import create_engine
engine = create_engine('postgresql+psycopg2://mikejcarey:postquel@localh
ost/mikejcarey')
```

In [ ]:
```
!cat /Users/mikejcarey/desktop/teaching/STATS170ab/TweetSecrets.txt
```

In [19]:
```
CONSUMER_KEY = "xxx"
CONSUMER_SECRET = "xxx"
ACCESS_KEY = "xxx-xxx"
ACCESS_SECRET = "xxx"
```

In [20]:
```
import tweepy

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_KEY, ACCESS_SECRET)

api = tweepy.API(auth)
```

In [21]:
```
# https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/
tweet-object
```

In [ ]:
```
# Here's how one can utilize the streaming API to gather a set of US geo
-located Tweets
'''
# Override tweepy.StreamListener to add logic to on_status

class MyStreamListener(tweepy.StreamListener):

    def __init__(self, tweets, target = 100, api=None):
        tweepy.StreamListener.__init__(self,api)
        self.coll_count=0
        self.tweets=tweets
        self.target_count=target

    def on_status(self, status):
        self.coll_count += 1
        if self.coll_count > self.target_count:
            return False
        if self.coll_count % 100 == 0:
            print(self.coll_count)
        self.tweets.append(status)

    def on_error(self, status_code):
        if status_code == 420:
            #returning False in on_data disconnects the stream
            return False

geotweets = [];
myStreamListener = MyStreamListener(geotweets,10000)
myStream = tweepy.Stream(auth = api.auth, listener=myStreamListener,
                         filter_level='medium', language='en')
myStream.filter(locations=[-125.0011,24.9493,-66.9669,49.5904]), async=T
rue)
'''
```

In [22]:
```
results = []
for tweet in tweepy.Cursor(api.search,
                           q="Winter Olympics medal",
                           include_entities=True,
                           wait_on_rate_limit=True,
                           wait_on_rate_limit_notify=True,
                           geocode="39.8,-95.583068847656,2500km",
                           lang="en").items(10000):
    results.append(tweet);
```

```
Rate limit reached. Sleeping for: 847
Rate limit reached. Sleeping for: 847
Rate limit reached. Sleeping for: 843
```

In [23]:
```
public_tweets = results
```

In [24]:
```
len(public_tweets)
```

Out[24]:
```
10000
```

In [26]:
```
# Here's the direct-to-PostgreSQL approach for storing the tweets
for t in public_tweets:
    values = (t.user.id_str, t.user.name, t.user.screen_name, t.user.des
cription, t.user.created_at)
    engine.execute("INSERT INTO Users (id, name, screen_name, descriptio
n, created_at) \
                    VALUES (%s, %s, %s, %s, %s) \
                    ON CONFLICT (id) DO UPDATE \
                    SET id = EXCLUDED.id, name = EXCLUDED.name, screen_n
ame = EXCLUDED.screen_name, \
                    description = EXCLUDED.description, created_at = EXC
LUDED.created_at", values)
    values = (t.created_at, t.id_str, t.text, t.user.id_str, t.in_reply_
to_status_id, t.in_reply_to_screen_name)
    engine.execute("INSERT INTO Tweets (created_at, id, text, user_id, i
n_reply_to_status_id, in_reply_to_screen_name) \
                    VALUES (%s, %s, %s, %s, %s, %s)", values)
    values = (t.id_str, json.dumps(t._json))
    engine.execute("INSERT INTO RawTweets (tweet_id, rawtweet) \
                    VALUES (%s, %s)", values)
    values = (t.id_str, t.user.id_str, t.user.statuses_count, t.user.fol
lowers_count, t.user.friends_count)
    engine.execute("INSERT INTO UserStats (tweet_id, user_id, statuses_c
ount, followers_count, friends_count) \
                    VALUES (%s, %s, %s, %s, %s) \
                    ON CONFLICT (user_id) DO UPDATE \
                    SET tweet_id = EXCLUDED.tweet_id, user_id = EXCLUDE
D.user_id, \
                    statuses_count = EXCLUDED.statuses_count, followers_
count = EXCLUDED.followers_count, \
                    friends_count = EXCLUDED.friends_count", values)
    for ht in t.entities['hashtags']:
        values = (t.id_str, ht['text'])
        engine.execute("INSERT INTO Hashtags (tweet_id, text) VALUES (%s
, %s)", values)
```

In [25]:
```
'''
engine.execute("DELETE FROM Hashtags")
engine.execute("DELETE FROM RawTweets")
engine.execute("DELETE FROM UserStats")
engine.execute("DELETE FROM Tweets")
engine.execute("DELETE FROM Users")
'''
```

Out[25]: <sqlalchemy.engine.result.ResultProxy at 0x10e4cc7f0>