Student Information

Please provide information about yourself.
Name1:
Name2 (optional) :
NetID 1:
NetID 2 (optional):
Notes to Grader (optional):

IMPORTANT Your work will not be graded withour your initials below
I certify that this lab represents my own work and I have read the RU academic intergrity policies at
https://www.cs.rutgers.edu/academic-integrity/introduction (https://www.cs.rutgers.edu/academic-integrity/introduction)
Initials:

Grader Notes

Your Grade:
Grader Initials:
Grader Comments (optional):

CS 439 - Introduction to Data Science

Fall 2019

Midsemester Project : Twitter Analysis

Due Date : November 4th, 2019 by 11:59 PM.

About this project

This is your mid-semester project. You are allowed to work with a partner (if you'd like). Only one partner (or max two per group is allowed). The goal of this mid-semester project is to work with Twitter API to analyze tweets from a person, and in this case, President Donald Trump. @RealDonaldTrump tweets provide a great opportunity to understand how online media can be used to communicate over the traditional media. Moreoover, Trump tweets has become so consequential, they actually can move the stock market on short term and get network TV to debate and discuss hours and hours about what Trump meant.
We hope this project will be fun as we can analyze range of emotions, hope, controversy, vagueness that are part of Trump tweets. We are interested in seeing what conclusions you can draw from US Presidents tweets.

- **DISCLAIMER: This project is not designed with any bias in mind. Note that we could pick either candiadate (Hillary Clinton or Donald Trump) or anyone else to do the same analysis. We hope your analysis is objective, independent of any political bias you may have. As Data Scientists, it is our responsiblity to do independent analysis of the data we try to understand. You should follow data and interpret w/o any bias.**

Set up

Let us get all the libaries initialized as necessary

In [1]:

```
# Run this cell to set up your notebook
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import zipfile
import json

# Ensure that Pandas shows at least 280 characters in columns, so we can see ful
l tweets
pd.set_option('max_colwidth', 280)

%matplotlib inline
plt.style.use('fivethirtyeight')
import seaborn as sns
sns.set()
sns.set_context("talk")
import re
```

Downloading Recent Tweets

It is important to download the most recent tweets (especially if you are working as a group). Those who are working by themselves are allowed to use the downloaded files w/o setting up access to any twitter API (which can sometime be bit complicated). Twitter provides the API Tweepy (http://www.tweepy.org/ (http://www.tweepy.org/)) that makes it easy to access twitter content that is publicly available. We will also provide example code as needed.

In [2]:

```
## Make sure you have set up tweepy if you are working locally.
# https://www.pythoncentral.io/introduction-to-tweepy-twitter-for-python/
# After set up, the following should run:
import tweepy
```

PART 1: Accessing Twitter API (optional for individuals)

In order to access Twitter API, you need to get keys by signing up as a Twitter developer. We will walk you through this process.

- **if you are working by yourself on this project, you can skip PART 1, and complete the project using the data files provided in the data folder. PART 1 is optional for those working by themselves. However, we highly recommend that you do Part 1 (after completing the project with offline data) if you would like to "learn" how to use Twitter API that might be useful.**

Task 1.1

Follow the instructions below to get your Twitter API keys. Read the instructions completely before starting.

1. Create a Twitter account (https://twitter.com/). You can use an existing account if you have one; if you prefer to not do this assignment under your regular account, feel free to create a throw-away account.
2. Under account settings, add your phone number to the account.
3. Create a Twitter developer account (https://developer.twitter.com/en/apply/) by clicking the 'Apply' button on the top right of the page. Attach it to your Twitter account. You'll have to fill out a form describing what you want to do with the developer account. Explain that you are doing this for a class at Rutgers University and that you don't know exactly what you're building yet and just need the account to get started. These applications are approved by some sort of AI system, so it doesn't matter exactly what you write. Just don't enter a bunch of alweiofalwiuhflawiuehflawuihflaiwhfe type stuff or you might get rejected.
4. Once you're logged into your developer account, create an application for this assignment (https://apps.twitter.com/app/new). You can call it whatever you want, and you can write any URL when it asks for a web site. You don't need to provide a callback URL.
5. On the page for that application, find your Consumer Key and Consumer Secret.
6. On the same page, create an Access Token. Record the resulting Access Token and Access Token Secret.
7. Edit the file keys.json (keys.json) and replace the placeholders with your keys.

WARNING (Please Read) !!!!

Protect your Twitter Keys

If someone has your authentication keys, they can access your Twitter account and post as you! So don't give them to anyone, and **don't write them down in this notebook**. The usual way to store sensitive information like this is to put it in a separate file and read it programmatically. That way, you can share the rest of your code without sharing your keys. That's why we're asking you to put your keys in `keys.json` for this assignment.

Avoid making too many API calls.

Twitter limits developers to a certain rate of requests for data. If you make too many requests in a short period of time, you'll have to wait awhile (around 15 minutes) before you can make more. So carefully follow the code examples you see and don't rerun cells without thinking. Instead, always save the data you've collected to a file. We've provided templates to help you do that.

Be careful about which functions you call!

This API can retweet tweets, follow and unfollow people, and modify your twitter settings. Be careful which functions you invoke! It is possible that you can accidentally re-tweet some tweets because you typed `retweet` instead of `retweet_count`. </span>

In [3]:

```
import json
key_file = 'keys.json'
# Loading your keys from keys.json (which you should have filled
# in in question 1):
with open(key_file) as f:
    keys = json.load(f)
# if you print or view the contents of keys be sure to delete the cell!
```

Task 1.2 Testing Twitter Authentication

This following code should run w/o erros or warnings and display yur twitter username. If you are working as a team, try to get a new twitter account with user names as : student1NetId_Student2NetId (eg: ds1089_adg133)

In [4]:

```
import tweepy
from tweepy import TweepError
import logging

try:
    auth = tweepy.OAuthHandler(keys["consumer_key"], keys["consumer_secret"])
    auth.set_access_token(keys["access_token"], keys["access_token_secret"])
    api = tweepy.API(auth)
    print("Your username is:", api.auth.get_username())
except TweepError as e:
    logging.warning("There was a Tweepy error. Double check your API keys and tr
y again.")
    logging.warning(e)
```

```
Your username is: cynthia_liqin
```

PART 2 - Working with Twitter

The json file in data folder contains (to be downloaded by you) some loaded tweets from @RutgersU. Run it and read the code. You can also try other json files in the data folder to try this.

In [5]:

```
from pathlib import Path
import json

ds_tweets_save_path = "data/RutgersU_recent_tweets.json"   # need to get this fi
le

# Guarding against attempts to download the data multiple
# times:
if not Path(ds_tweets_save_path).is_file():
    # Getting as many recent tweets by @RutgersU as Twitter will let us have.
    # We use tweet_mode='extended' so that Twitter gives us full 280 character t
weets.
    # This was a change introduced in September 2017.

    # The tweepy Cursor API actually returns "sophisticated" Status objects but
 we
    # will use the basic Python dictionaries stored in the _json field.
    example_tweets = [t._json for t in tweepy.Cursor(api.user_timeline, id="Rutg
ersU",
                                                     tweet_mode='extended').items()]

    # Saving the tweets to a json file on disk for future analysis
    with open(ds_tweets_save_path, "w") as f:
        json.dump(example_tweets, f)

# Re-loading the json file:
with open(ds_tweets_save_path, "r") as f:
    example_tweets = json.load(f)
```

If things ran as expected, you should be able to look at the first tweet by running the code below. It probabably does not make sense to view all tweets in a notebook, as size of the tweets can freeze your browser (always a good idea to press ctrl-S to save the latest, in case you have to restart Jupyter)

In [6]:

```
# Looking at one tweet object, which has type Status:
from pprint import pprint # ...to get a more easily-readable view.
pprint(example_tweets[0])
```

```
{'contributors': None,
 'coordinates': None,
 'created_at': 'Wed Oct 16 17:23:28 +0000 2019',
 'display_text_range': [0, 140],
 'entities': {'hashtags': [],
              'symbols': [],
              'urls': [],
              'user_mentions': [{'id': 30223211,
                                 'id_str': '30223211',
                                 'indices': [3, 15],
                                 'name': 'Rutgers Scarlet Knights',
                                 'screen_name': 'RUAthletics'},
                                {'id': 818431566,
                                 'id_str': '818431566',
                                 'indices': [17, 27],
                                 'name': 'The Birthplace',
                                 'screen_name': 'RFootball'},
                                {'id': 2249043324,
                                 'id_str': '2249043324',
                                 'indices': [28, 42],
                                 'name': 'adidas Football US',
                                 'screen_name': 'adidasFballUS'},
                                {'id': 61312712,
                                 'id_str': '61312712',
                                 'indices': [43, 52],
                                 'name': 'adidas',
                                 'screen_name': 'adidasUS'},
                                {'id': 19272796,
                                 'id_str': '19272796',
                                 'indices': [53, 62],
                                 'name': 'Rutgers University',
                                 'screen_name': 'RutgersU'},
                                {'id': 879068137,
                                 'id_str': '879068137',
                                 'indices': [63, 71],
                                 'name': 'UNISWAG',
                                 'screen_name': 'UNISWAG'},
                                {'id': 955836661702774784,
                                 'id_str': '955836661702774784',
                                 'indices': [72, 82],
                                 'name': 'Rutgers University–New Bru
nswick',
                                 'screen_name': 'RutgersNB'},
                                {'id': 211321066,
                                 'id_str': '211321066',
                                 'indices': [83, 94],
                                 'name': 'Phil Hecken',
                                 'screen_name': 'PhilHecken'},
                                {'id': 7768692,
                                 'id_str': '7768692',
                                 'indices': [95, 109],
                                 'name': 'Big Ten Network',
                                 'screen_name': 'BigTenNetwork'},
                                {'id': 348590880,
                                 'id_str': '348590880',
                                 'indices': [110, 119],
                                 'name': 'FOX College Football',
                                 'screen_name': 'CFBONFOX'}]},
 'favorite_count': 0,
 'favorited': False,
 'full_text': 'RT @RUAthletics: @RFootball @adidasFballUS @adidasUS
```

```
@RutgersU '
              '@UNISWAG @RutgersNB @PhilHecken @BigTenNetwork @CFBON
FOX What '
              'does it mean t…',
 'geo': None,
 'id': 1184520243454107653,
 'id_str': '1184520243454107653',
 'in_reply_to_screen_name': None,
 'in_reply_to_status_id': None,
 'in_reply_to_status_id_str': None,
 'in_reply_to_user_id': None,
 'in_reply_to_user_id_str': None,
 'is_quote_status': False,
 'lang': 'en',
 'place': None,
 'retweet_count': 9,
 'retweeted': False,
 'retweeted_status': {'contributors': None,
                      'coordinates': None,
                      'created_at': 'Wed Oct 16 15:54:39 +0000 201
9',
                      'display_text_range': [103, 341],
                      'entities': {'hashtags': [{'indices': [315, 32
9],
                                                 'text': 'TheBirthpl
ace'}],
                                   'media': [{'display_url': 'pic.tw
itter.com/L8O7CHWBXY',
                                              'expanded_url': 'http
s://twitter.com/RUAthletics/status/1184497892783988737/photo/1',
                                              'id': 1184497881836871
680,
                                              'id_str': '11844978818
36871680',
                                              'indices': [342, 365],
                                              'media_url': 'http://p
bs.twimg.com/media/EHAujDwW4AAlI8E.jpg',
                                              'media_url_https': 'ht
tps://pbs.twimg.com/media/EHAujDwW4AAlI8E.jpg',
                                              'sizes': {'large':
{'h': 1366,
                                                                   'r
esize': 'fit',

'w': 2048},
                                                        'medium':
{'h': 800,

'resize': 'fit',

'w': 1200},
                                                        'small':
{'h': 453,
                                                                   'r
esize': 'fit',

'w': 680},
                                                        'thumb':
{'h': 150,
                                                                   'r
esize': 'crop',
'w': 150}},
                                              'type': 'photo',
                                              'url': 'https://t.co/L
8O7CHWBXY'}],
                                   'symbols': [],
                                   'urls': [],
                                   'user_mentions': [{'id': 81843156
6,
                                                     'id_str': '818
431566',
                                                     'indices': [0,
10],
                                                     'name': 'The B
irthplace',
                                                     'screen_name':
'RFootball'},
                                                    {'id': 22490433
24,
                                                     'id_str': '224
9043324',
                                                     'indices': [1
1, 25],
                                                     'name': 'adida
s Football '
                                                            'US',
                                                     'screen_name':
'adidasFballUS'},
                                                    {'id': 6131271
2,
                                                     'id_str': '613
12712',
                                                     'indices': [2
6, 35],
                                                     'name': 'adida
s',
                                                     'screen_name':
'adidasUS'},
                                                    {'id': 1927279
6,
                                                     'id_str': '192
72796',
                                                     'indices': [3
6, 45],
                                                     'name': 'Rutge
rs '
                                                            'Unive
rsity',
                                                     'screen_name':
'RutgersU'},
                                                    {'id': 87906813
7,
                                                     'id_str': '879
068137',
                                                     'indices': [4
6, 54],
                                                     'name': 'UNISW
AG',
                                                     'screen_name':
'UNISWAG'},
                                                    {'id': 95583666
```

```
1702774784,
                                                    'id_str': '955
836661702774784',
                                                    'indices': [5
5, 65],
                                                    'name': 'Rutge
rs '
                                                            'Unive
rsity–New '
                                                            'Bruns
wick',
                                                    'screen_name':
'RutgersNB'},
                                                   {'id': 21132106
6,
                                                    'id_str': '211
321066',
                                                    'indices': [6
6, 77],
                                                    'name': 'Phil
Hecken',
                                                    'screen_name':
'PhilHecken'},
                                                   {'id': 7768692,
                                                    'id_str': '776
8692',
                                                    'indices': [7
8, 92],
                                                    'name': 'Big T
en Network',
                                                    'screen_name':
'BigTenNetwork'},
                                                   {'id': 34859088
0,
                                                    'id_str': '348
590880',
                                                    'indices': [9
3, 102],
                                                    'name': 'FOX C
ollege '
                                                            'Footb
all',
                                                    'screen_name':
'CFBONFOX'},
                                                   {'id': 81843156
6,
                                                    'id_str': '818
431566',
                                                    'indices': [13
4, 144],
                                                    'name': 'The B
irthplace',
                                                    'screen_name':
'RFootball'}]},
                      'extended_entities': {'media': [{'display_ur
l': 'pic.twitter.com/L8O7CHWBXY',
                                                    'expanded_ur
l': 'https://twitter.com/RUAthletics/status/1184497892783988737/phot
o/1',
                                                    'id': 1184497
881836871680,
                                                    'id_str': '11
84497881836871680',
                                                    'indices': [3
42, 365],
                                                    'media_url':
'http://pbs.twimg.com/media/EHAujDwW4AAlI8E.jpg',
                                                    'media_url_ht
tps': 'https://pbs.twimg.com/media/EHAujDwW4AAlI8E.jpg',
                                                    'sizes': {'la
rge': {'h': 1366,

'resize': 'fit',

'w': 2048},
                                                            'me
dium': {'h': 800,

'resize': 'fit',

'w': 1200},
                                                            'sm
all': {'h': 453,

'resize': 'fit',

'w': 680},
                                                            'th
umb': {'h': 150,

'resize': 'crop',

'w': 150}},
                                                    'type': 'phot
o',
                                                    'url': 'http
s://t.co/L8O7CHWBXY'},
                                                   {'display_ur
l': 'pic.twitter.com/L8O7CHWBXY',
                                                    'expanded_ur
l': 'https://twitter.com/RUAthletics/status/1184497892783988737/phot
o/1',
                                                    'id': 1184497
881836900353,
                                                    'id_str': '11
84497881836900353',
                                                    'indices': [3
42, 365],
                                                    'media_url':
'http://pbs.twimg.com/media/EHAujDwXUAEGQkA.jpg',
                                                    'media_url_ht
tps': 'https://pbs.twimg.com/media/EHAujDwXUAEGQkA.jpg',
                                                    'sizes': {'la
rge': {'h': 1366,

'resize': 'fit',

'w': 2048},
                                                            'me
dium': {'h': 800,

'resize': 'fit',
'w': 1200},
                                                                  'sm
all': {'h': 453,

'resize': 'fit',

'w': 680},
                                                                  'th
umb': {'h': 150,

'resize': 'crop',

'w': 150}},
                                                         'type': 'phot
o',
                                                         'url': 'http
s://t.co/L8O7CHWBXY'},
                                                        {'display_ur
l': 'pic.twitter.com/L8O7CHWBXY',
                                                         'expanded_ur
l': 'https://twitter.com/RUAthletics/status/1184497892783988737/phot
o/1',
                                                         'id': 1184497
881845243904,
                                                         'id_str': '11
84497881845243904',
                                                         'indices': [3
42, 365],
                                                         'media_url':
'http://pbs.twimg.com/media/EHAujDyWoAAblNb.jpg',
                                                         'media_url_ht
tps': 'https://pbs.twimg.com/media/EHAujDyWoAAblNb.jpg',
                                                         'sizes': {'la
rge': {'h': 1366,

'resize': 'fit',

'w': 2048},
                                                                  'me
dium': {'h': 800,

'resize': 'fit',

'w': 1200},
                                                                  'sm
all': {'h': 453,

'resize': 'fit',

'w': 680},
                                                                  'th
umb': {'h': 150,

'resize': 'crop',

'w': 150}},
                                                         'type': 'phot
o',
                                                         'url': 'http
s://t.co/L8O7CHWBXY'}]},
                      'favorite_count': 26,
                      'favorited': False,
                      'full_text': '@RFootball @adidasFballUS @adida
sUS '
                                   '@RutgersU @UNISWAG @RutgersNB @P
hilHecken '
                                   '@BigTenNetwork @CFBONFOX What do
es it mean '
                                   'to be first?\n'
                                   '@RFootball knows. \n'
                                   '\n'
                                   'We tap into our roots this Satur
day by '
                                   'donning long-sleeve uniforms, th
e first '
                                   'team in the modern era to do so,
at the '
                                   'place where it all began 150 yea
rs ago. \n'
                                   '\n'
                                   '#TheBirthplace | Est. 1869 '
                                   'https://t.co/L8O7CHWBXY',
                      'geo': None,
                      'id': 1184497892783988737,
                      'id_str': '1184497892783988737',
                      'in_reply_to_screen_name': 'RUAthletics',
                      'in_reply_to_status_id': 1184119164329431041,
                      'in_reply_to_status_id_str': '1184119164329431
041',
                      'in_reply_to_user_id': 30223211,
                      'in_reply_to_user_id_str': '30223211',
                      'is_quote_status': False,
                      'lang': 'en',
                      'place': None,
                      'possibly_sensitive': False,
                      'retweet_count': 9,
                      'retweeted': False,
                      'source': '<a href="https://mobile.twitter.co
m" '
                                'rel="nofollow">Twitter Web App</a
>',
                      'truncated': False,
                      'user': {'contributors_enabled': False,
                               'created_at': 'Fri Apr 10 14:15:49 +0
000 2009',
                               'default_profile': False,
                               'default_profile_image': False,
                               'description': 'In New Jersey, our fa
ns are '
                                              'just louder 🔊 #GoRU |
'
                                              '#JerseyPride l #ShowY
ourR l ⚔️🛡',
                               'entities': {'description': {'urls':
[]},
                                            'url': {'urls': [{'displ
ay_url': 'scarletknights.com',
                                                              'expan
ded_url': 'http://www.scarletknights.com',
                                                              'indic
es': [0,
23],
                                                                   'url':
'https://t.co/IbLc4EF7Jt'}]}},
                               'favourites_count': 15300,
                               'follow_request_sent': False,
                               'followers_count': 63542,
                               'following': False,
                               'friends_count': 546,
                               'geo_enabled': True,
                               'has_extended_profile': False,
                               'id': 30223211,
                               'id_str': '30223211',
                               'is_translation_enabled': False,
                               'is_translator': False,
                               'lang': None,
                               'listed_count': 528,
                               'location': 'The Birthplace of Colleg
e Football',
                               'name': 'Rutgers Scarlet Knights',
                               'notifications': False,
                               'profile_background_color': '131516',
                               'profile_background_image_url': 'htt
p://abs.twimg.com/images/themes/theme14/bg.gif',
                               'profile_background_image_url_https':
'https://abs.twimg.com/images/themes/theme14/bg.gif',
                               'profile_background_tile': True,
                               'profile_banner_url': 'https://pbs.tw
img.com/profile_banners/30223211/1559821354',
                               'profile_image_url': 'http://pbs.twim
g.com/profile_images/973215306054283264/4kh4PDbt_normal.jpg',
                               'profile_image_url_https': 'https://p
bs.twimg.com/profile_images/973215306054283264/4kh4PDbt_normal.jpg',
                               'profile_link_color': 'D41B1B',
                               'profile_sidebar_border_color': 'FFFF
FF',
                               'profile_sidebar_fill_color': 'EFEFE
F',
                               'profile_text_color': '333333',
                               'profile_use_background_image': True,
                               'protected': False,
                               'screen_name': 'RUAthletics',
                               'statuses_count': 34010,
                               'time_zone': None,
                               'translator_type': 'none',
                               'url': 'https://t.co/IbLc4EF7Jt',
                               'utc_offset': None,
                               'verified': True}},
 'source': '<a href="https://mobile.twitter.com" rel="nofollow">Twit
ter Web '
           'App</a>',
 'truncated': False,
 'user': {'contributors_enabled': False,
          'created_at': 'Wed Jan 21 02:57:47 +0000 2009',
          'default_profile': False,
          'default_profile_image': False,
          'description': 'Rutgers, The State University of New Jerse
y, is a '
                         'leading public research university. Follow
us for '
                         'all things Rutgers.',
          'entities': {'description': {'urls': []},
                       'url': {'urls': [{'display_url': 'rutgers.ed
u',
                                         'expanded_url': 'http://ww
w.rutgers.edu',
                                         'indices': [0, 22],
                                         'url': 'http://t.co/stAPJIz
h8b'}]}},
          'favourites_count': 4505,
          'follow_request_sent': False,
          'followers_count': 132672,
          'following': True,
          'friends_count': 593,
          'geo_enabled': True,
          'has_extended_profile': False,
          'id': 19272796,
          'id_str': '19272796',
          'is_translation_enabled': False,
          'is_translator': False,
          'lang': None,
          'listed_count': 841,
          'location': 'New Jersey',
          'name': 'Rutgers University',
          'notifications': False,
          'profile_background_color': 'C7141C',
          'profile_background_image_url': 'http://abs.twimg.com/imag
es/themes/theme15/bg.png',
          'profile_background_image_url_https': 'https://abs.twimg.c
om/images/themes/theme15/bg.png',
          'profile_background_tile': False,
          'profile_banner_url': 'https://pbs.twimg.com/profile_banne
rs/19272796/1494779773',
          'profile_image_url': 'http://pbs.twimg.com/profile_images/
809450270375772160/rWmyBIig_normal.jpg',
          'profile_image_url_https': 'https://pbs.twimg.com/profile_
images/809450270375772160/rWmyBIig_normal.jpg',
          'profile_link_color': '0084B4',
          'profile_sidebar_border_color': '000205',
          'profile_sidebar_fill_color': 'C0DFEC',
          'profile_text_color': '333333',
          'profile_use_background_image': False,
          'protected': False,
          'screen_name': 'RutgersU',
          'statuses_count': 16003,
          'time_zone': None,
          'translator_type': 'none',
          'url': 'http://t.co/stAPJIzh8b',
          'utc_offset': None,
          'verified': True}}
```

Task 2.1 (Optional for Individuals)

What you need to do.

Re-factor the above code fragment into reusable snippets below. You should not need to make major modifications; this is mostly an exercise in understanding the above code block.

In [7]:

```
def load_keys(path):
    """Loads your Twitter authentication keys from a file on disk.

    Args:
        path (str): The path to your key file.  The file should
          be in JSON format and look like this (but filled in):
            {
                "consumer_key": "<your Consumer Key here>",
                "consumer_secret":  "<your Consumer Secret here>",
                "access_token": "<your Access Token here>",
                "access_token_secret": "<your Access Token Secret here>"
            }

    Returns:
        dict: A dictionary mapping key names (like "consumer_key") to
          key values."""

    ### BEGIN SOLUTION

    # your solution here

    ### END SOLUTION
```

In [8]:

```
def download_recent_tweets_by_user(user_account_name, keys):
    """Downloads tweets by one Twitter user.

    Args:
        user_account_name (str): The name of the Twitter account
          whose tweets will be downloaded.
        keys (dict): A Python dictionary with Twitter authentication
          keys (strings), like this (but filled in):
            {
                "consumer_key": "<your Consumer Key here>",
                "consumer_secret":  "<your Consumer Secret here>",
                "access_token": "<your Access Token here>",
                "access_token_secret": "<your Access Token Secret here>"
            }

    Returns:
        list: A list of Dictonary objects, each representing one tweet."""
    import tweepy

    ### BEGIN SOLUTION

    # your solution here

    ### END SOLUTION
```

In [9]:

```
def load_tweets(path):
    """Loads tweets that have previously been saved.

    Calling load_tweets(path) after save_tweets(tweets, path)
    will produce the same list of tweets.

    Args:
        path (str): The place where the tweets were be saved.

    Returns:
        list: A list of Dictionary objects, each representing one tweet."""

    ### BEGIN SOLUTION

    # your solution here

    ### END SOLUTION
```

In [10]:

```
def get_tweets_with_cache(user_account_name, keys_path):
    """Get recent tweets from one user, loading from a disk cache if available.

    The first time you call this function, it will download tweets by
    a user.  Subsequent calls will not re-download the tweets; instead
    they'll load the tweets from a save file in your local filesystem.
    All this is done using the functions you defined in the previous cell.
    This has benefits and drawbacks that often appear when you cache data:

    +: Using this function will prevent extraneous usage of the Twitter API.
    +: You will get your data much faster after the first time it's called.
    -: If you really want to re-download the tweets (say, to get newer ones,
       or because you screwed up something in the previous cell and your
       tweets aren't what you wanted), you'll have to find the save file
       (which will look like <something>_recent_tweets.pkl) and delete it.

    Args:
        user_account_name (str): The Twitter handle of a user, without the @.
        keys_path (str): The path to a JSON keys file in your filesystem.
    """

    ### BEGIN SOLUTION

    # your solution here

    ### END SOLUTION
```

If everything was implemented correctly you should be able to obtain roughly the last 3000 tweets by the realdonaldtrump. (This may take a few minutes)

In [11]:

```
# When you are done, run this cell to load @realdonaldtrump's tweets.
# Note the function get_tweets_with_cache.  You may find it useful
# later.
trump_tweets = get_tweets_with_cache("realdonaldtrump", key_file)
print("Number of tweets downloaded:", len(trump_tweets))
```

```
Number of tweets downloaded: 200
```

Task 2.2

To be consistent we are going to use the same dataset no matter what you get from your twitter api. So from this point on, if you are working as a group or individually, be sure to use the data sets provided to you in the zip file. There should be two json files inside your data folder. One is '2017-2018.json', the other one is '2016-2017.json'. We will load the '2017-2018.json' first.

In [3]:

```
def load_tweets(path):
    """Loads tweets that have previously been saved.

    Calling load_tweets(path) after save_tweets(tweets, path)
    will produce the same list of tweets.

    Args:
        path (str): The place where the tweets were be saved.

    Returns:
        list: A list of Dictionary objects, each representing one tweet."""

    with open(path, "rb") as f:
        import json
        return json.load(f)
```

In [7]:

```
dest_path = 'data/2017-2018.json'
trump_tweets = load_tweets(dest_path)
```

If everything is working correctly correctly this should load roughly the last 3000 tweets by `realdonaldtrump`.

In [8]:

```
assert 2000 <= len(trump_tweets) <= 4000
```

If the assert statement above works, then continue on to task 2.3.

Task 2.3

Find the number of the month of the oldest tweet.

In [9]:

```
# Enter the number of the month of the oldest tweet (e.g. 1 for January)
oldest_month = 8

### BEGIN SOLUTION

#*** code to compute ****
### END SOLUTION
```

Out[9]:

```
2071    4
Name: created_at, dtype: int64
```

PART 3 Twitter Source Analysis

Task 3.1

Create a new date frame named `all_tweets` from `2016-2017.json` and merge with `trump_tweets`

Important: There may/will be some overlap so be sure to eliminate duplicate tweets. If you do not eliminate the duplicates properly, your results might not be compatible with the test solution. Hint: the `id` of a tweet is always unique.

In [12]:

```
# if you do not have new tweets, then all_tweets is the same as  old_trump_tweet
s

### BEGIN SOLUTION

### END SOLUTION
print(all_tweets.size)
assert(all_tweets.size == 40176)
```

```
40176
```

Task 3.2

Construct a DataFrame called `df_trump` containing all the tweets stored in `all_tweets`. The index of the dataframe should be the ID of each tweet (looks something like `907698529606541312`). It should have these columns:

- **`time`: The time the tweet was created encoded as a datetime object. (Use `pd.to_datetime` to encode the timestamp.)**
- **`source`: The source device of the tweet.**
- **`text`: The text of the tweet.**
- **`retweet_count`: The retweet count of the tweet.**

Finally, the resulting dataframe should be sorted by the index.

Warning: ***Some tweets will store the text in the `text` field and other will use the `full_text` field.***

Warning: ***Don't forget to check the type of index***

In [13]:

```
### BEGIN SOLUTION

### END SOLUTION
```

Out[13]:

	time	source	text	retweet_count
id				
682723973449289728	2016-01-01 00:44:14+00:00	Twitter for Android	I will be on @FoxNews live, with members of my family, at 11:50 P.M. We will ring in the New Year together! MAKE AMERICA GREAT AGAIN!	2108
682764544402440192	2016-01-01 03:25:27+00:00	Twitter for iPhone	HAPPY NEW YEAR &amp; THANK YOU! https://t.co/YO1Yi8QbZy https://t.co/uxUXWJ1Rbv	3460
682792967736848385	2016-01-01 05:18:23+00:00	Twitter for iPhone	#HappyNewYearAmerica! https://t.co/EeQb8PDrUe	3434
682805320217980929	2016-01-01 06:07:28+00:00	Twitter for iPhone	Happy New Year from #MarALago! Thank you to my great family for all of their support. https://t.co/6UsqSiaaj7	1948
682805477168779264	2016-01-01 06:08:06+00:00	Twitter for Android	"@jallenaip: Hillary said she was in a "Fog of War" as explanation for the lies about Benghazi. No fog allowed in WH. Vote Trump POTUS!"	2721

In the following questions, we are going to find out the charateristics of Trump tweets and the devices used for the tweets.

First let's examine the source field:

In [14]:

```
df_trump['source'].unique()
```

Out[14]:

```
array(['Twitter for Android', 'Twitter for iPhone', 'Twitter Web Cli
ent',
       'Mobile Web (M5)', 'Instagram', 'Twitter Ads', 'Twitter for i
Pad',
       'Media Studio', 'TweetDeck', 'Periscope',
       '<a href="http://twitter.com/download/iphone" rel="nofollow">
Twitter for iPhone</a>',
       '<a href="https://studio.twitter.com" rel="nofollow">Media St
udio</a>',
       '<a href="http://twitter.com/#!/download/ipad" rel="nofollo
w">Twitter for iPad</a>',
       '<a href="http://twitter.com" rel="nofollow">Twitter Web Clie
nt</a>'],
      dtype=object)
```

Task 3.3

Remove the HTML tags from the source field.

Hint: Use `df_trump['source'].str.replace` and your favorite regular expression.

In [15]:

```
### BEGIN SOLUTION

### END SOLUTION
```

Out[15]:

```
id
682723973449289728     Twitter for Android
682764544402440192      Twitter for iPhone
682792967736848385      Twitter for iPhone
682805320217980929      Twitter for iPhone
682805477168779264     Twitter for Android
                              ...
1052213711295930368     Twitter for iPhone
1052217314463100928     Twitter for iPhone
1052219253384994816     Twitter for iPhone
1052232230972678145     Twitter for iPhone
1052233253040640001     Twitter for iPhone
Name: source, Length: 10044, dtype: object
```

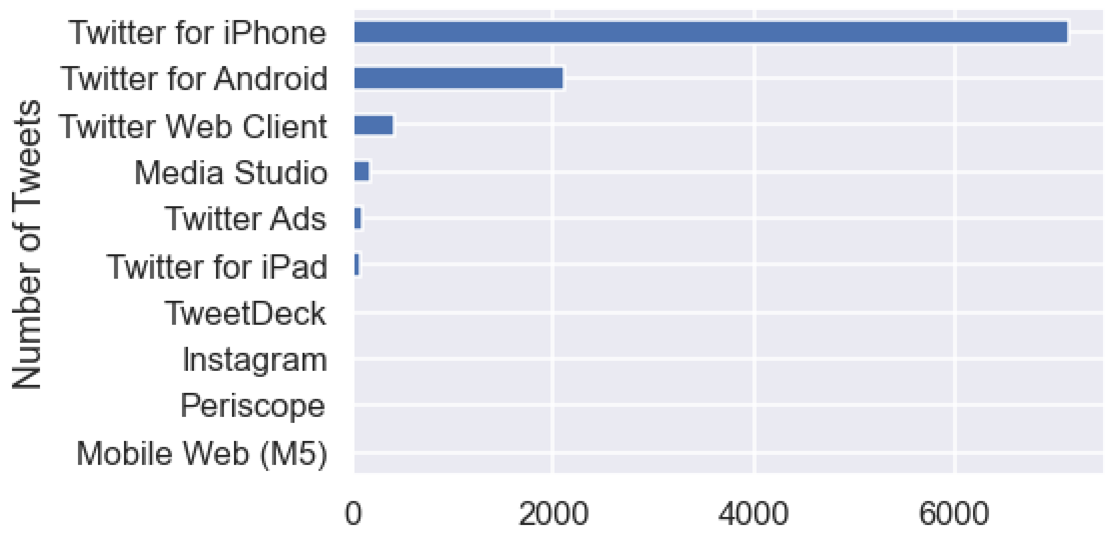
Make a plot to find out the most common device types used in accessing twitter

In [16]:

```
### BEGIN SOLUTION

### END SOLUTION
```

Out[16]:

```
Text(0, 0.5, 'Number of Tweets')
```



Task 3.4

Is there a difference between his Tweet behavior across these devices? We will attempt to answer this question in our subsequent analysis.

First, we'll take a look at whether Trump's tweets from an Android come at different times than his tweets from an iPhone. Note that Twitter gives us his tweets in the [UTC timezone (https://www.wikiwand.com/en/List_of_UTC_time_offsets)](https://www.wikiwand.com/en/List_of_UTC_time_offsets) (notice the `+0000` in the first few tweets)

In [17]:

```
df_trump['time'][0:3]
```

Out[17]:

```
id
682723973449289728   2016-01-01 00:44:14+00:00
682764544402440192   2016-01-01 03:25:27+00:00
682792967736848385   2016-01-01 05:18:23+00:00
Name: time, dtype: datetime64[ns, UTC]
```

We'll convert the tweet times to US Eastern Time, the timezone of New York and Washington D.C., since those are the places we would expect the most tweet activity from Trump.

In [18]:

```
df_trump['est_time'] = (
    df_trump['time'] # Set initial timezone to UTC
                 .dt.tz_convert("EST") # Convert to Eastern Time
)
df_trump.head()
```

Out[18]:

	time	source	text	retweet_count	est_tin
id					
682723973449289728	2016-01-01 00:44:14+00:00	Twitter for Android	I will be on @FoxNews live, with members of my family, at 11:50 P.M. We will ring in the New Year together! MAKE AMERICA GREAT AGAIN!	2108	2015-1 19:44:1 05:
682764544402440192	2016-01-01 03:25:27+00:00	Twitter for iPhone	HAPPY NEW YEAR &amp; THANK YOU! https://t.co/YO1Yi8QbZy https://t.co/uxUXWJ1Rbv	3460	2015-1 22:25:2 05:
682792967736848385	2016-01-01 05:18:23+00:00	Twitter for iPhone	#HappyNewYearAmerica! https://t.co/EeQb8PDrUe	3434	2016-0 00:18:2 05:
682805320217980929	2016-01-01 06:07:28+00:00	Twitter for iPhone	Happy New Year from #MarALago! Thank you to my great family for all of their support. https://t.co/6UsqSiaaj7	1948	2016-0 01:07:2 05:
682805477168779264	2016-01-01 06:08:06+00:00	Twitter for Android	"@jallenaip: Hillary said she was in a "Fog of War" as explanation for the lies about Benghazi. No fog allowed in WH. Vote Trump POTUS!"	2721	2016-0 01:08:0 05:

What you need to do:

Add a column called `hour` to the `df_trump` table which contains the hour of the day as floating point number computed by:

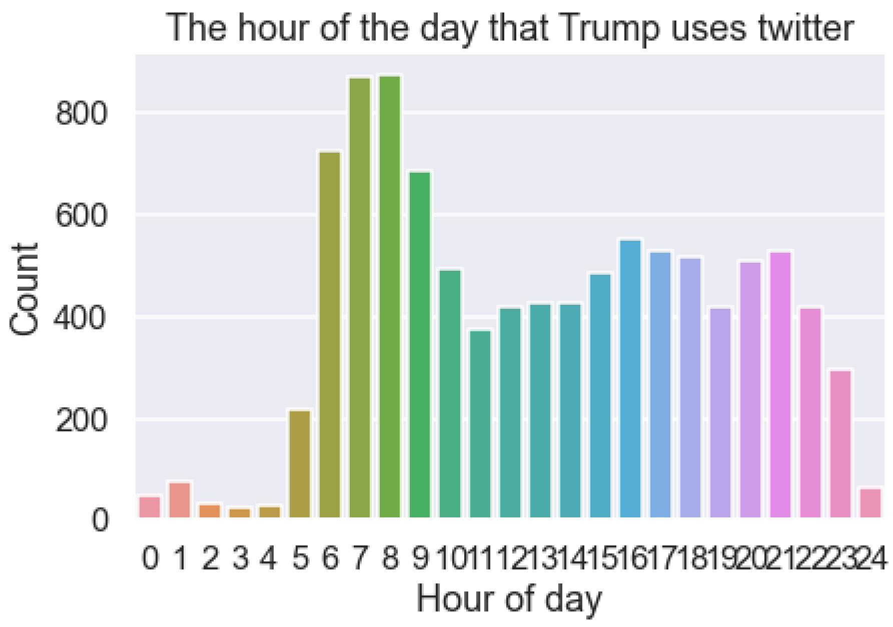
$$\text{hour} + \frac{\text{minute}}{60} + \frac{\text{second}}{60^2}$$

In [19]:

```
df_trump['hour'] = df_trump.est_time.apply(lambda x: x.hour + x.minute/60 + x.se
cond/3600)
df_trump['roundhour']=round(df_trump['hour'])
# make a bar plot here
### BEGIN SOLUTION


### END SOLUTION
```

Out[19]:

```
Text(0.5, 1.0, 'The hour of the day that Trump uses twitter')
```



In [20]:

```
assert np.isclose(df_trump.loc["690171032150237184"]['hour'], 8.93639)
```

Use this data along with the seaborn `distplot` function to examine the distribution over hours of the day in eastern time that trump tweets on each device for the 2 most commonly used devices.

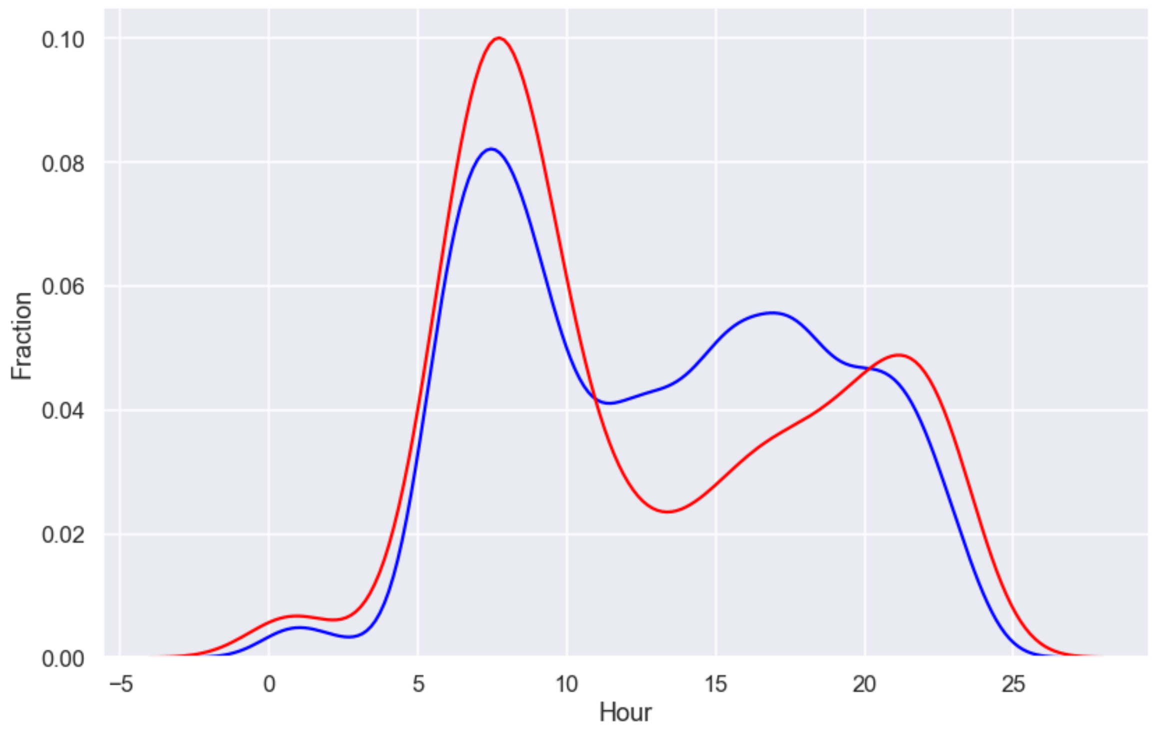
In [21]:

```
### BEGIN SOLUTION
### make your plot here

### END SOLUTION
```

```
/Users/dd/miniconda3/lib/python3.8/site-packages/seaborn/distributio
ns.py:2551: FutureWarning: `distplot` is a deprecated function and w
ill be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility) or
`kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
/Users/dd/miniconda3/lib/python3.8/site-packages/seaborn/distributio
ns.py:2551: FutureWarning: `distplot` is a deprecated function and w
ill be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility) or
`kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```

Out[21]:

```
Text(0, 0.5, 'Fraction')
```



Task 3.5

According to this Verge article (https://www.theverge.com/2017/3/29/15103504/donald-trump-iphone-using-switched-android), Donald Trump switched from an Android to an iPhone sometime in March 2017.

Create a figure identical to your figure from 3.4, except that you should show the results only from 2016. If you get stuck consider looking at the `year_fraction` function from the next problem.

Use this data along with the seaborn `distplot` function to examine the distribution over hours of the day in eastern time that trump tweets on each device for the 2 most commonly used devices. Your plot should look somewhat similar to the following.

During the campaign, it was theorized that Donald Trump's tweets from Android were written by him personally, and the tweets from iPhone were from his staff. Does your figure give support the theory?

Response: In 2016, the time allocation for the usage of the iphone centered in the afternoon, while his tweets from 2015 to present shows that he mostly tweets in the morning. It seems that the tweets from iphone in 2016 were from his staff, not himself.

title

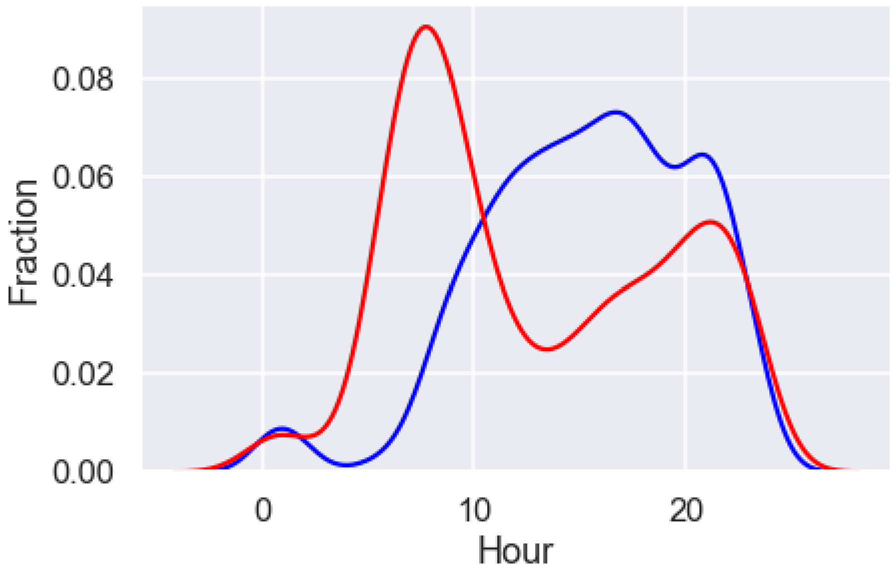
In [22]:

```
### BEGIN SOLUTION

# code to plot

### END SOLUTION
```

```
/Users/dd/miniconda3/lib/python3.8/site-packages/seaborn/distributio
ns.py:2551: FutureWarning: `distplot` is a deprecated function and w
ill be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility) or
`kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
/Users/dd/miniconda3/lib/python3.8/site-packages/seaborn/distributio
ns.py:2551: FutureWarning: `distplot` is a deprecated function and w
ill be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility) or
`kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```

Out[22]:

```
Text(0, 0.5, 'Fraction')
```



Task 3.6

Edit this cell to answer the following questions.

- **What time of the day the Android tweets were made by Trump himself? (eg: morning, late night etc)**
- **What time of the day the Android tweets were made by paid staff?**

Note that these are speculations based on what you observe in the data set.

Task 3.7 Device Analysis

Let's now look at which device he has used over the entire time period of this dataset.

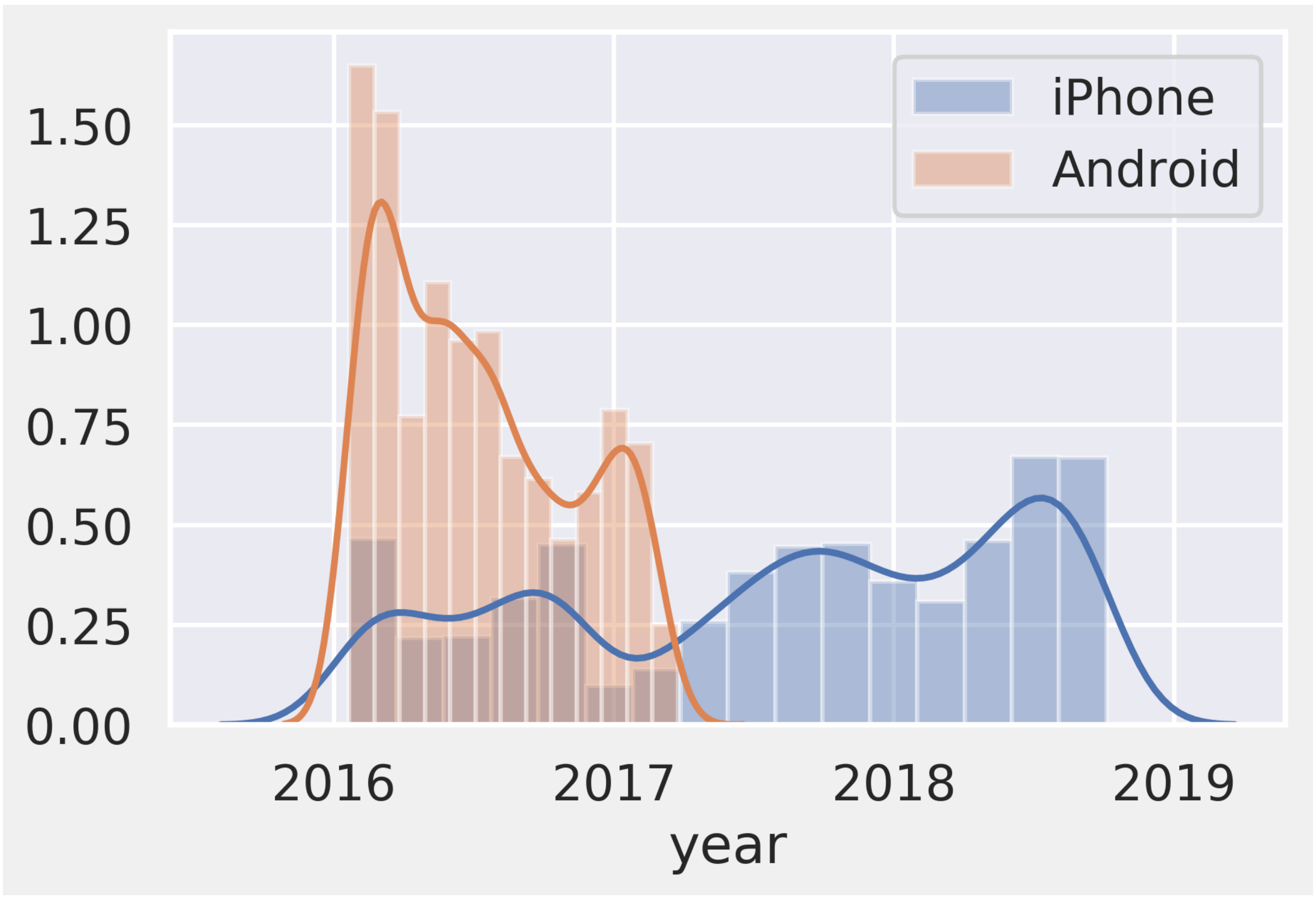
To examine the distribution of dates we will convert the date to a fractional year that can be plotted as a distribution.

(Code borrowed from https://stackoverflow.com/questions/6451655/python-how-to-convert-datetime-dates-to-decimal-years (https://stackoverflow.com/questions/6451655/python-how-to-convert-datetime-dates-to-decimal-years))

In [23]:

```
import datetime
def year_fraction(date):
    start = datetime.date(date.year, 1, 1).toordinal()
    year_length = datetime.date(date.year+1, 1, 1).toordinal() - start
    return date.year + float(date.toordinal() - start) / year_length


df_trump['year'] = df_trump['time'].apply(year_fraction) #should be df_trump
```

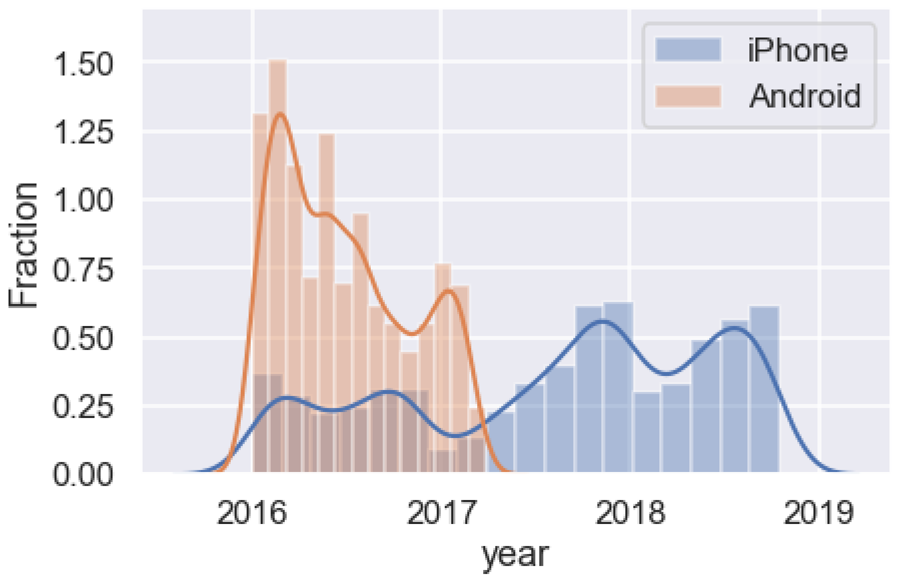
Use the `sns.distplot` to overlay the distributions of the 2 most frequently used web technologies over the years. Your final plot should be similar to:



In [24]:

```
### BEGIN SOLUTION


### END SOLUTION
```

```
/Users/dd/miniconda3/lib/python3.8/site-packages/seaborn/distributio
ns.py:2551: FutureWarning: `distplot` is a deprecated function and w
ill be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility) or
`histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/Users/dd/miniconda3/lib/python3.8/site-packages/seaborn/distributio
ns.py:2551: FutureWarning: `distplot` is a deprecated function and w
ill be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility) or
`histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



PART 4 - Sentiment Analysis

It turns out that we can use the words in Trump's tweets to calculate a measure of the sentiment of the tweet. For example, the sentence "I love America!" has positive sentiment, whereas the sentence "I hate taxes!" has a negative sentiment. In addition, some words have stronger positive / negative sentiment than others: "I love America." is more positive than "I like America."

We will use the [VADER (Valence Aware Dictionary and sEntiment Reasoner) (https://github.com/cjhutto/vaderSentiment)](https://github.com/cjhutto/vaderSentiment) lexicon to analyze the sentiment of Trump's tweets. VADER is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media which is great for our usage.

The VADER lexicon gives the sentiment of individual words. Run the following cell to show the first few rows of the lexicon:

In [26]:

```
print(''.join(open("data/vader_lexicon.txt").readlines()[:10]))
```

```
$:      -1.5    0.80623 [-1, -1, -1, -1, -3, -1, -3, -1, -2, -1]
%)      -0.4    1.0198  [-1, 0, -1, 0, 0, -2, -1, 2, -1, 0]
%-)     -1.5    1.43178 [-2, 0, -2, -2, -1, 2, -2, -3, -2, -3]
&-:     -0.4    1.42829 [-3, -1, 0, 0, -1, -1, -1, 2, -1, 2]
&:      -0.7    0.64031 [0, -1, -1, -1, 1, -1, -1, -1, -1, -1]
( '}{' )        1.6     0.66332 [1, 2, 2, 1, 1, 2, 2, 1, 3, 1]
(%      -0.9    0.9434  [0, 0, 1, -1, -1, -1, -2, -2, -1, -2]
('-:    2.2     1.16619 [4, 1, 4, 3, 1, 2, 3, 1, 2, 1]
(':     2.3     0.9     [1, 3, 3, 2, 2, 4, 2, 3, 1, 2]
((-:    2.1     0.53852 [2, 2, 2, 1, 2, 3, 2, 2, 3, 2]
```

Task 4.1

As you can see, the lexicon contains emojis too! The first column of the lexicon is the *token*, or the word itself. The second column is the *polarity* of the word, or how positive / negative it is.

(How did they decide the polarities of these words? What are the other two columns in the lexicon? See the link above.)

Read in the lexicon into a DataFrame called `df_sent` . The index of the DF should be the tokens in the lexicon. `df_sent` should have one column: `polarity` : The polarity of each token.

In [27]:

```
### BEGIN SOLUTION

### END SOLUTION
```

Task 4.2

Now, let's use this lexicon to calculate the overall sentiment for each of Trump's tweets. Here's the basic idea:

1. For each tweet, find the sentiment of each word.
2. Calculate the sentiment of each tweet by taking the sum of the sentiments of its words.

First, let's lowercase the text in the tweets since the lexicon is also lowercase. Set the `text` column of the `df_trump` DF to be the lowercased text of each tweet.

In [28]:

```
### BEGIN SOLUTION

### END SOLUTION
```

Task 4.3

Now, let's get rid of punctuation since it'll cause us to fail to match words. Create a new column called `no_punc` in the `df_trump` to be the lowercased text of each tweet with all punctuation replaced by a single space. We consider punctuation characters to be any character that isn't a Unicode word character or a whitespace character. You may want to consult the Python documentation on regexes for this problem.

(Why don't we simply remove punctuation instead of replacing with a space? See if you can figure this out by looking at the tweet data.)

In [29]:

```
# Save your regex in punct_re
punct_re = r'[^\w\s\\n]'


### BEGIN SOLUTION

### END SOLUTION
```

In [30]:

```
assert isinstance(punct_re, str)
assert re.search(punct_re, 'this') is None
assert re.search(punct_re, 'this is ok') is None
assert re.search(punct_re, 'this is\nok') is None
assert re.search(punct_re, 'this is not ok.') is not None
assert re.search(punct_re, 'this#is#ok') is not None
assert re.search(punct_re, 'this^is ok') is not None
assert df_trump['no_punc'].loc[800329364986626048] == 'i watched parts of  nbcsn
l saturday night live last night  it is a totally one sided  biased show   nothi
ng funny at all  equal time for us '
assert df_trump['text'].loc[884740553040175104] == 'working hard to get the olym
pics for the united states (l.a.). stay tuned!'
```

Task 4.4

Now, let's convert the tweets into what's called a *tidy format* (https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html) to make the sentiments easier to calculate. Use the `no_punc` column of `df_trump` to create a table called `tidy_format`. The index of the table should be the IDs of the tweets, repeated once for every word in the tweet. It has two columns:

1. **`num`: The location of the word in the tweet. For example, if the tweet was "i love america", then the location of the word "i" is 0, "love" is 1, and "america" is 2.**
2. **`word`: The individual words of each tweet.**

The first few rows of our `tidy_format` table look like:

	num	word
894661651760377856	0	i
894661651760377856	1	think
894661651760377856	2	senator
894661651760377856	3	blumenthal
894661651760377856	4	should

Note that you'll get different results depending on when you pulled in the tweets. However, you can double check that your tweet with ID `894661651760377856` has the same rows as ours. Our tests don't check whether your table looks exactly like ours.

As usual, try to avoid using any for loops. Our solution uses a chain of 5 methods on the 'trump' DF, albeit using some rather advanced Pandas hacking.

- **Hint 1: Try looking at the `expand` argument to pandas' `str.split`.**
- **Hint 2: Try looking at the `stack()` method.**
- **Hint 3: Try looking at the `level` parameter of the `reset_index` method.**

In [31]:

```
#tidy_format = ...

### BEGIN SOLUTION

### END SOLUTION
```

Out[31]:

	num	word
682723973449289728	0	i
682723973449289728	1	will
682723973449289728	2	be
682723973449289728	3	on
682723973449289728	4	foxnews

In [32]:

```
assert tidy_format.loc[894661651760377856].shape == (27, 2)
assert ' '.join(list(tidy_format.loc[894661651760377856]['word'])) == 'i think s
enator blumenthal should take a nice long vacation in vietnam where he lied abou
t his service so he can at least say he was there'
```

Task 4.5

Now that we have this table in the tidy format, it becomes much easier to find the sentiment of each tweet: we can join the table with the lexicon table.

Add a `polarity` column to the `trump` table. The `polarity` column should contain the sum of the sentiment polarity of each word in the text of the tweet.

Hint you will need to merge the `tidy_format` and `sent` tables and group the final answer.

In [33]:

```
#df_trump['polarity'] = ...

### BEGIN SOLUTION
### END SOLUTION
```

Out[33]:

	time	source	text	retweet_count	est_ti
682723973449289728	2016-01-01 00:44:14+00:00	Twitter for Android	i will be on @foxnews live, with members of my family, at 11:50 p.m. we will ring in the new year together! make america great again!	2108	2015- 19:44: 05:
682764544402440192	2016-01-01 03:25:27+00:00	Twitter for iPhone	happy new year &amp; thank you! https://t.co/yo1yi8qbzy https://t.co/uxuxwj1rbv	3460	2015- 22:25: 05:
682792967736848385	2016-01-01 05:18:23+00:00	Twitter for iPhone	#happynewyearamerica! https://t.co/eeqb8pdrue	3434	2016- 00:18: 05:
682805320217980929	2016-01-01 06:07:28+00:00	Twitter for iPhone	happy new year from #maralago! thank you to my great family for all of their support. https://t.co/6usqsiaaj7	1948	2016- 01:07: 05:
682805477168779264	2016-01-01 06:08:06+00:00	Twitter for Android	"@jallenaip: hillary said she was in a "fog of war" as explanation for the lies about benghazi. no fog allowed in wh. vote trump potus!"	2721	2016- 01:08: 05:
...	...	...	...	...	
1052213711295930368	2018-10-16 15:04:32+00:00	Twitter for iPhone	“federal judge throws out stormy danials lawsuit versus trump. trump is entitled to full legal fees.” @foxnews great, now i can go after horseface and her 3rd rate lawyer in the great state of texas. she will confirm the letter she signed! she knows nothing about me, a total ...	14594	2018- 10:04: 05:
1052217314463100928	2018-10-16 15:18:51+00:00	Twitter for iPhone	“conflict between glen simpson’s testimony to another house panel about his contact with justice department official bruce ohr. ohr was used by simpson and steele as a back channel to get (fake) dossier to fbi. simpson pleading fifth.” catherine herridge. where is jeff sessions?	6271	2018- 10:18: 05:

	time	source	text	retweet_count	est_ti
1052219253384994816	2018-10-16 15:26:33+00:00	Twitter for iPhone	is it really possible that bruce ohr, whose wife nellie was paid by simpson and gps fusion for work done on the fake dossier, and who was used as a pawn in this whole scam (witch hunt), is still working for the department of justice????? can this really be so?????	13103	2018-1 10:26:3 05:
1052232230972678145	2018-10-16 16:18:08+00:00	Twitter for iPhone	rt @whitehouse: https://t.co/rnqlpots3o	4478	2018-1 11:18:0 05:
1052233253040640001	2018-10-16 16:22:11+00:00	Twitter for iPhone	register to https://t.co/0pwiwchgbh! #maga🇺🇸 https://t.co/actme53tzu	5415	2018-1 11:22:1 05:

10044 rows × 10 columns

In [34]:

```
assert np.allclose(df_trump.loc[744701872456536064, 'polarity'], 8.4)
assert np.allclose(df_trump.loc[745304731346702336, 'polarity'], 2.5)
assert np.allclose(df_trump.loc[744519497764184064, 'polarity'], 1.7)
assert np.allclose(df_trump.loc[894661651760377856, 'polarity'], 0.2)
assert np.allclose(df_trump.loc[894620077634592769, 'polarity'], 5.4)
# If you fail this test, you dropped tweets with 0 polarity
#assert np.allclose(df_trump.loc[744355251365511169, 'polarity'], 0.0)
```

Task 4.6

Now we have a measure of the sentiment of each of his tweets! You can read over the VADER readme to understand a more robust sentiment analysis. Now, write the code to see the most positive and most negative tweets from Trump in your dataset: Find the most negative and most positive tweets made by Trump

In [35]:

```
### BEGIN SOLUTION

### END SOLUTION
```

```
Most negative tweets:

   horrible and cowardly terrorist attack on innocent and defenseles
s worshipers in egypt. the world cannot tolerate terrorism, we must
defeat them militarily and discredit the extremist ideology that for
ms the basis of their existence!

   horrible and cowardly terrorist attack on innocent and defenseles
s worshipers in egypt. the world cannot tolerate terrorism, we must
defeat them militarily and discredit the extremist ideology that for
ms the basis of their existence!

   nyc terrorist was happy as he asked to hang isis flag in his hosp
ital room. he killed 8 people, badly injured 12. should get death pe
nalty!

   nyc terrorist was happy as he asked to hang isis flag in his hosp
ital room. he killed 8 people, badly injured 12. should get death pe
nalty!

   fake news cnn made a vicious and purposeful mistake yesterday. th
ey were caught red handed, just like lonely brian ross at abc news
(who should be immediately fired for his “mistake”). watch to see if
@cnn fires those responsible, or was it just gross incompetence?
```

In [36]:

```
### BEGIN SOLUTION

### END SOLUTION
```

```
Most positive tweets:

   it was my great honor to celebrate the opening of two extraordina
ry museums-the mississippi state history museum &amp; the mississipp
i civil rights museum. we pay solemn tribute to our heroes of the pa
st &amp; dedicate ourselves to building a future of freedom, equalit
y, justice &amp; peace. https://t.co/5akgvpv8aa

   it was my great honor to celebrate the opening of two extraordina
ry museums-the mississippi state history museum &amp; the mississipp
i civil rights museum. we pay solemn tribute to our heroes of the pa
st &amp; dedicate ourselves to building a future of freedom, equalit
y, justice &amp; peace. https://t.co/5akgvpv8aa

   melania, our great and very hard working first lady, who truly lo
ves what she is doing, always thought that “if you run, you will wi
n.” she would tell everyone that, “no doubt, he will win.” i also fe
lt i would win (or i would not have run) - and country is doing grea
t!

   melania, our great and very hard working first lady, who truly lo
ves what she is doing, always thought that “if you run, you will wi
n.” she would tell everyone that, “no doubt, he will win.” i also fe
lt i would win (or i would not have run) - and country is doing grea
t!

   throughout my travels, i've had the pleasure of sharing the good
news from america. i've had the honor of sharing our vision for a fr
ee &amp; open indo-pacific -- a place where sovereign &amp; independ
ent nations, w/diverse cultures &amp; many different dreams, can all
prosper side-by-side. https://t.co/qbocy3u7yv
```

Task 4.7

Plot the distribution of tweet sentiments broken down by whether the text of the tweet contains `nyt` or `fox` . Then in the box below comment on what we observe?

title

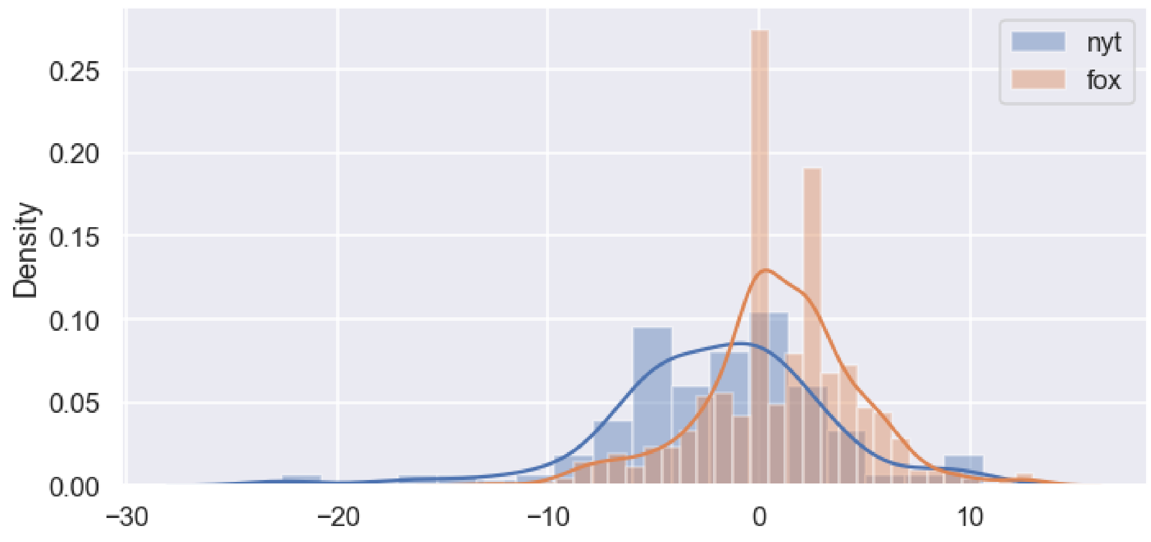
In [37]:

```
### BEGIN SOLUTION
### END SOLUTION
```

```
/Users/dd/miniconda3/lib/python3.8/site-packages/seaborn/distributio
ns.py:2551: FutureWarning: `distplot` is a deprecated function and w
ill be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility) or
`histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/Users/dd/miniconda3/lib/python3.8/site-packages/seaborn/distributio
ns.py:2551: FutureWarning: `distplot` is a deprecated function and w
ill be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility) or
`histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[37]:

```
<matplotlib.legend.Legend at 0x12d270430>
```



Comment on what you observe:

BEGIN SOLUTION

END SOLUTION

PART 5 - Principal Component Analysis (PCA) and Twitter

A look at the top words used and the sentiments expressed in Trump tweets indicates that, some words are used with others almost all the time. A notable example is the slogan like Make America Great Again. As such, it may be beneficial to look at groups of words rather than individual words. For that, we will look at an approach applying a Principal Component Analysis.

The PCA

The Principal Component Analysis, or PCA, is a tool generally used to identify patterns and to reduce the number of variables you have to consider in your analysis. For example, if you have data with 200 columns, it may be that a significant amount of the variance in your data can be explained by just 100 principal components. In the PCA, the first component is chosen in such a way that has the largest variance, subsequent components are orthogonal and continue covering as much variance as possible. In this way, the PCA samples as much of the variability in the data set with the first few components. Mathematically, each component is a linear combination of all the input parameters times coefficients specific for that component. These coefficients, or loading factors, are constrained such that the sum of the squares of them are equal to 1. As such, the loading factors serve as weights describing how strongly certain parameters contribute to the specific principal component. Parameters with large values of positive or negative loading factors are correlated with each other, which can serve to identify trends in your data.

Task 5.1 Cleaning up the Data

Using NLTK (Natural Language Toolkit) package for language processing and other python libraries, parse the json file to deal with inflected words, such as plurals, and removed stop words like common English words (the, and, it, etc) and certain political terms (the candidates names, for example). You can start with the top 50 words, but full analysis may require large number of words. Create a document-freqeucy (df) matrix with 5000 rows and 50 columns where each column is a particular word (feature) and each row is a tweet (observation). The values of the matrix is how often the word appears. Apply the techniques we learned to reduce the weight of most common words (if necessary). Since this is a sparse matrix, you can use the sparse martix libraries to make things a bit more efficient (we can also use a regular numpy arrays to store these things since the dimensions are not too large). Lecture 6.1 captures some sparse matrix routines you can use. Print the first 10 rows of the df to show the matrix you created

In [39]:

```
!pip install nltk
```

```
Collecting nltk
  Downloading nltk-3.5.zip (1.4 MB)
     |████████████████████████████████| 1.4 MB 4.5 MB/s eta 0:00:01
Collecting click
  Downloading click-7.1.2-py2.py3-none-any.whl (82 kB)
     |████████████████████████████████| 82 kB 3.8 MB/s  eta 0:00:01
Collecting joblib
  Downloading joblib-0.17.0-py3-none-any.whl (301 kB)
     |████████████████████████████████| 301 kB 25.8 MB/s eta 0:00:01
Collecting regex
  Downloading regex-2020.10.23.tar.gz (690 kB)
     |████████████████████████████████| 690 kB 21.7 MB/s eta 0:00:01
Requirement already satisfied: tqdm in /Users/dd/miniconda3/lib/pyth
on3.8/site-packages (from nltk) (4.46.0)
Building wheels for collected packages: nltk, regex
  Building wheel for nltk (setup.py) ... done
  Created wheel for nltk: filename=nltk-3.5-py3-none-any.whl size=14
34678 sha256=a4f3da6518f5d34ef16db757d0827f645697d1be2969b7f8bfbf078
1ad8d7a5f
  Stored in directory: /Users/dd/Library/Caches/pip/wheels/ff/d5/7b/
f1fb4e1e1603b2f01c2424dd60fbcc50c12ef918bafc44b155
  Building wheel for regex (setup.py) ... done
  Created wheel for regex: filename=regex-2020.10.23-cp38-cp38-macos
x_10_9_x86_64.whl size=279924 sha256=cc3c611e51965f4adb49e19144c618a
42eebc4efe41206b82b3153ef34a42b5c
  Stored in directory: /Users/dd/Library/Caches/pip/wheels/a6/ac/74/
98324bc614b7706aaee1e7cf6e4f7563c43f09681fead0de1a
Successfully built nltk regex
Installing collected packages: click, joblib, regex, nltk
Successfully installed click-7.1.2 joblib-0.17.0 nltk-3.5 regex-202
0.10.23
```

In [40]:

```
### BEGIN SOLUTION

### END SOLUTION
```

```
[nltk_data] Downloading package stopwords to /Users/dd/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /Users/dd/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
```

Out[40]:

```
array([[1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
        0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
0.,
        0., 0.],
       [1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
0.,
        0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
        0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
0.,
        0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
1.,
        0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
0.,
        0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
0.,
        1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
        0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
0.,
        0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
0.,
        1., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
0.,
        0., 0.],
       [1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 0., 1., 0., 0.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
0.,
        0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
0.,
        0., 0.]])
```

Task 5.2 Find the PCA's

Write the code to find the first 50 PCA's for the document-frequency matrix. Pass the document-term-matrix to scikit-learn’s (https://scikit-learn.org/stable/modules/decomposition.html#decompositions (https://scikit-learn.org/stable/modules/decomposition.html#decompositions)) PCA method to obtain the components and loading factors.

In [42]:

```
!pip install sklearn
```

```
Collecting sklearn
  Downloading sklearn-0.0.tar.gz (1.1 kB)
Collecting scikit-learn
  Downloading scikit_learn-0.23.2-cp38-cp38-macosx_10_9_x86_64.whl
(7.2 MB)
     |████████████████████████████████| 7.2 MB 3.9 MB/s eta 0:00:01
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-2.1.0-py3-none-any.whl (12 kB)
Requirement already satisfied: numpy>=1.13.3 in /Users/dd/miniconda
3/lib/python3.8/site-packages (from scikit-learn->sklearn) (1.19.1)
Requirement already satisfied: scipy>=0.19.1 in /Users/dd/miniconda
3/lib/python3.8/site-packages (from scikit-learn->sklearn) (1.5.2)
Requirement already satisfied: joblib>=0.11 in /Users/dd/miniconda3/
lib/python3.8/site-packages (from scikit-learn->sklearn) (0.17.0)
Building wheels for collected packages: sklearn
  Building wheel for sklearn (setup.py) ... done
  Created wheel for sklearn: filename=sklearn-0.0-py2.py3-none-any.w
hl size=1315 sha256=529890287b089e21fbdec17c97efba164d313a757b67f49a
07bf39ecf2a94c7d
  Stored in directory: /Users/dd/Library/Caches/pip/wheels/22/0b/40/
fd3f795caaa1fb4c6cb738bc1f56100be1e57da95849bfc897
Successfully built sklearn
Installing collected packages: threadpoolctl, scikit-learn, sklearn
Successfully installed scikit-learn-0.23.2 sklearn-0.0 threadpoolctl
-2.1.0
```

In [43]:

```
### BEGIN SOLUTION
#Very useful website to learn: https://jakevdp.github.io/PythonDataScienceHandbo
ok/05.09-principal-component-analysis.html

### END SOLUTION
```

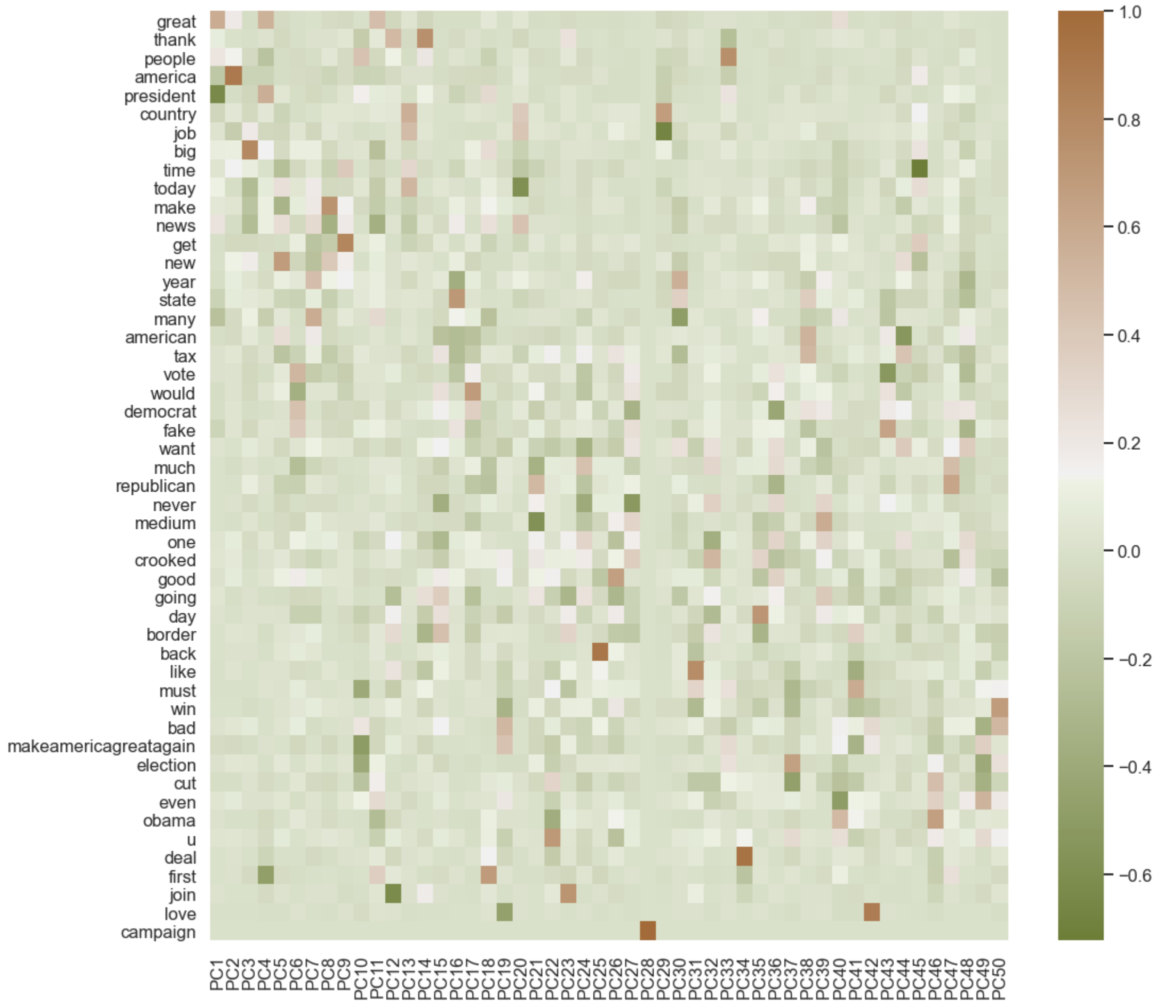
```
[[ 5.77997273e-01  1.87434417e-01 -2.27339132e-02 ... -1.36556113e-0
3
  -4.08220123e-03 -1.59204770e-02]
 [ 9.02192446e-02 -1.84869965e-03 -1.07226855e-02 ... -3.95448149e-0
2
   2.77456380e-03 -1.38499759e-02]
 [ 2.20151991e-01  1.38032919e-01 -4.72996901e-02 ...  6.49174084e-0
2
  -1.59864956e-03 -1.57922184e-02]
 ...
 [ 1.20460158e-02 -8.46187426e-04 -7.63461889e-04 ... -9.29184154e-0
3
  -4.74275768e-02  2.80447429e-02]
 [-3.21503221e-03 -4.55539625e-03 -6.53067670e-03 ...  1.83413463e-0
3
   2.03797306e-06 -1.81337971e-02]
 [-0.00000000e+00 -1.79922680e-17 -4.47003416e-17 ... -8.32667268e-1
7
  -7.28583860e-17  2.63677968e-16]]
[3.36447040e-01 1.60190465e-01 1.48864080e-01 1.40857697e-01
 1.16520716e-01 1.00415118e-01 9.39223022e-02 8.26574506e-02
 7.22602368e-02 6.97821140e-02 6.61198507e-02 6.50476268e-02
 6.32787025e-02 5.96806458e-02 5.27240038e-02 5.13413776e-02
 4.95646294e-02 4.65913669e-02 4.56887854e-02 4.29642620e-02
 4.28084435e-02 4.14735459e-02 4.03694481e-02 3.90642846e-02
 3.86992436e-02 3.67382069e-02 3.58504598e-02 3.56057808e-02
 3.48255733e-02 3.40251196e-02 3.34313726e-02 3.28237217e-02
 3.06711475e-02 3.01774624e-02 2.86315320e-02 2.77145112e-02
 2.65827924e-02 2.59463173e-02 2.55132025e-02 2.46143668e-02
 2.35032227e-02 2.26277740e-02 2.21539888e-02 2.10205514e-02
 2.07272941e-02 1.77152273e-02 1.62249736e-02 1.38486922e-02
 9.93092052e-03 3.46916650e-34]
```

Task 5.3 Examine the PCA

We can examine the PCA results to look at the heatmap. Make a grid plot which shows the various principal component along the x-axis and the individual words along the y-axes. Each grid box should be color-coded based on the sign of the loading factor and how large the square of that value is. Looking at it vertically, you can see which words constitute your principal components. Looking at it horizontally, you can see how individual terms are shared between components.

In [44]:

```
### BEGIN SOLUTION

### END SOLUTION
```



Task 5.4 PCA Compare

We can determine how many words and how many components are needed to do a good visualization. Plot PC1 and PC2 in a 2D plot. The results should be similar to following scatter plot

title

This is a scatter plot of the values of the components, but with arrows indicating some of the prominent terms as indicated by their loading factors. The values of the loading factors are used to determine the length and direction of these arrows and as such they serve as a way of expressing direction. That is, tweets which use these terms will be moved along the length of those arrows. Shown are the most important parameters.

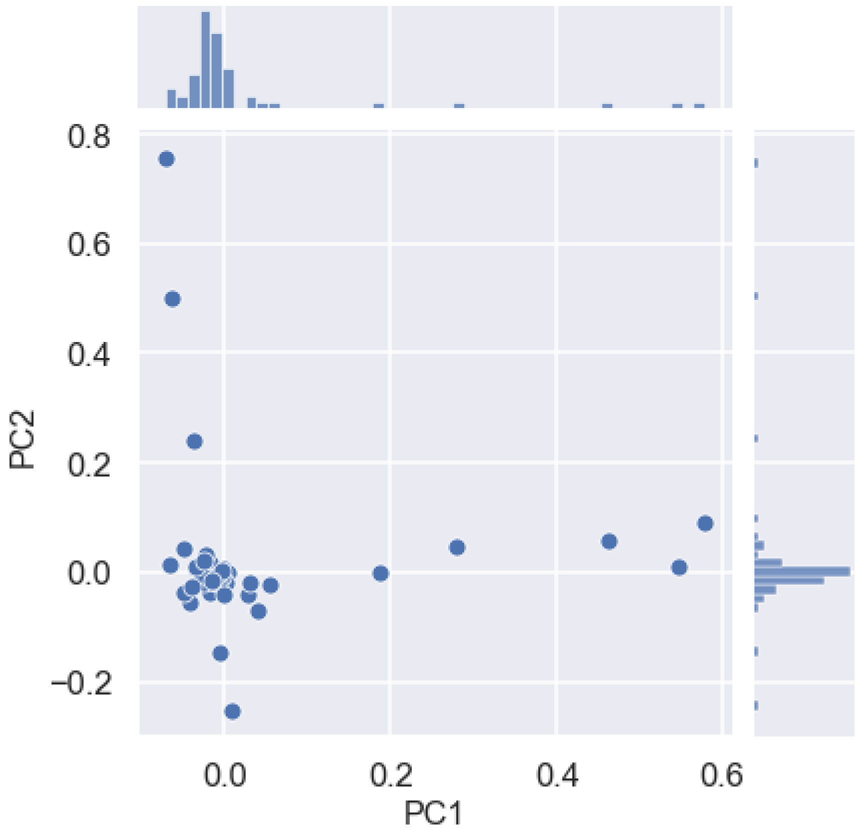
In [45]:

```
### BEGIN SOLUTION

### END SOLUTION
```

```
/Users/dd/miniconda3/lib/python3.8/site-packages/seaborn/_decorator
s.py:36: FutureWarning: Pass the following variables as keyword arg
s: x, y. From version 0.12, the only valid positional argument will
be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```

Out[45]:

```
<seaborn.axisgrid.JointGrid at 0x13ccdd7f0>
```



PART 6 - Twitter Engagement

In this problem, we'll explore which words led to a greater average number of retweets. For example, at the time of this writing, Donald Trump has two tweets that contain the word 'oakland' (tweets 932570628451954688 and 1016609920031117312) with 36757 and 10286 retweets respectively, for an average of 23,521.5.

Your `top_20` table should have this format:

	retweet_count
word	
jong	40675.666667
try	33937.800000
kim	32849.595745
un	32741.731707
maybe	30473.192308

Task 6.1

Find the top 20 most retweeted words. Include only words that appear in at least 25 tweets. As usual, try to do this without any for loops. You can string together ~7 pandas commands and get everything done on one line.

In [46]:

```
#top_20 = ...
### BEGIN SOLUTION
### END SOLUTION
```

Out[46]:

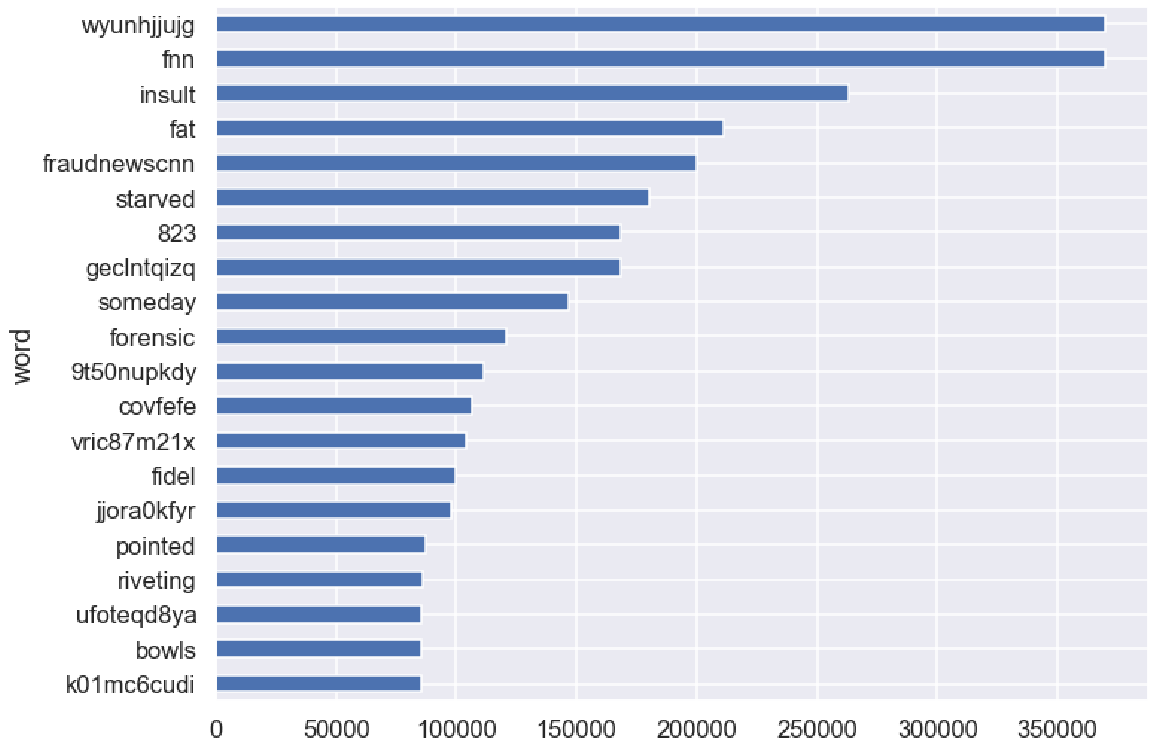
	retweet_count
word	
wyunhjjujg	369530.000
fnn	369530.000
insult	263388.000
fat	211133.600
fraudnewscnn	200188.500
starved	180010.000
823	168765.000
geclntqizq	168765.000
someday	147100.875
forensic	120661.000
9t50nupkdy	111303.000
covfefe	106531.000
vric87m21x	104295.000
fidel	99483.000
jjora0kfyr	97922.000
pointed	87163.000
riveting	86169.000
bowls	85465.000
ufoteqd8ya	85465.000
k01mc6cudi	85465.000

Task 6.2

Plot a bar chart of your results:

In [47]:

```
### BEGIN SOLUTION
### BEGIN SOLUTION
```



PART 7 - Conclusion (Optional for Individual)

What else can we do? Let us ask some open ended questions.

Task 7.1

"kim", "jong" and "un" are apparently really popular in Trump's tweets! It seems like we can conclude that his tweets involving jong are more popular than his other tweets. Or can we?

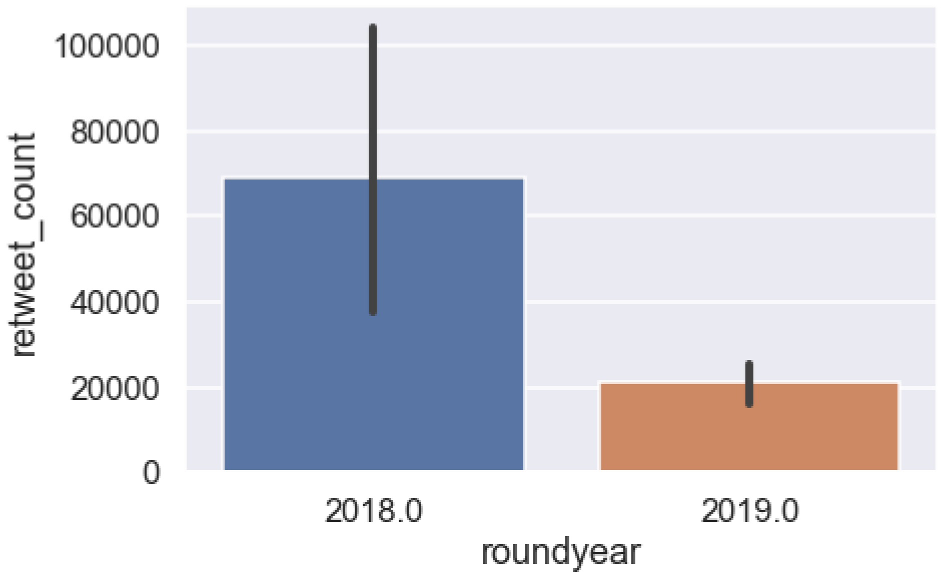
Consider each of the statements about possible confounding factors below. State whether each statement is true or false and explain. If the statement is true, state whether the confounding factor could have made kim jong un related tweets higher in the list than they should be.

1. **We didn't restrict our word list to nouns, so we have unhelpful words like "let" and "any" in our result.**
   - **That might be why 'un' is the most popular.**
2. **We didn't remove hashtags in our text, so we have duplicate words (eg. #great and great).**
   - **Some may only have '#great' not 'great' which make the average lower**
3. **We didn't account for the fact that Trump's follower count has increased over time.**
   - **This can affect a lot. As Trump's follower count has increased, the more popular every word be**

In [48]:

```
#### BEGIN SOLUTION
#### END SOLUTION
```

```
<ipython-input-48-b6f447545052>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py
  jong['roundyear']=round(jong['year'])
```



Task 7.2

Using the `df_trump` tweets construct an interesting plot describing a property of the data and discuss what you found below.

Ideas:

1. How has the sentiment changed with length of the tweets?
2. Does sentiment affect retweet count?
3. Are retweets more negative than regular tweets?
4. Are there any spikes in the number of retweets and do the correspond to world events?
5. What terms have an especially positive or negative sentiment?

You can look at other data sources and even tweets. Do some plots and discuss. You can add more cells here as needed.

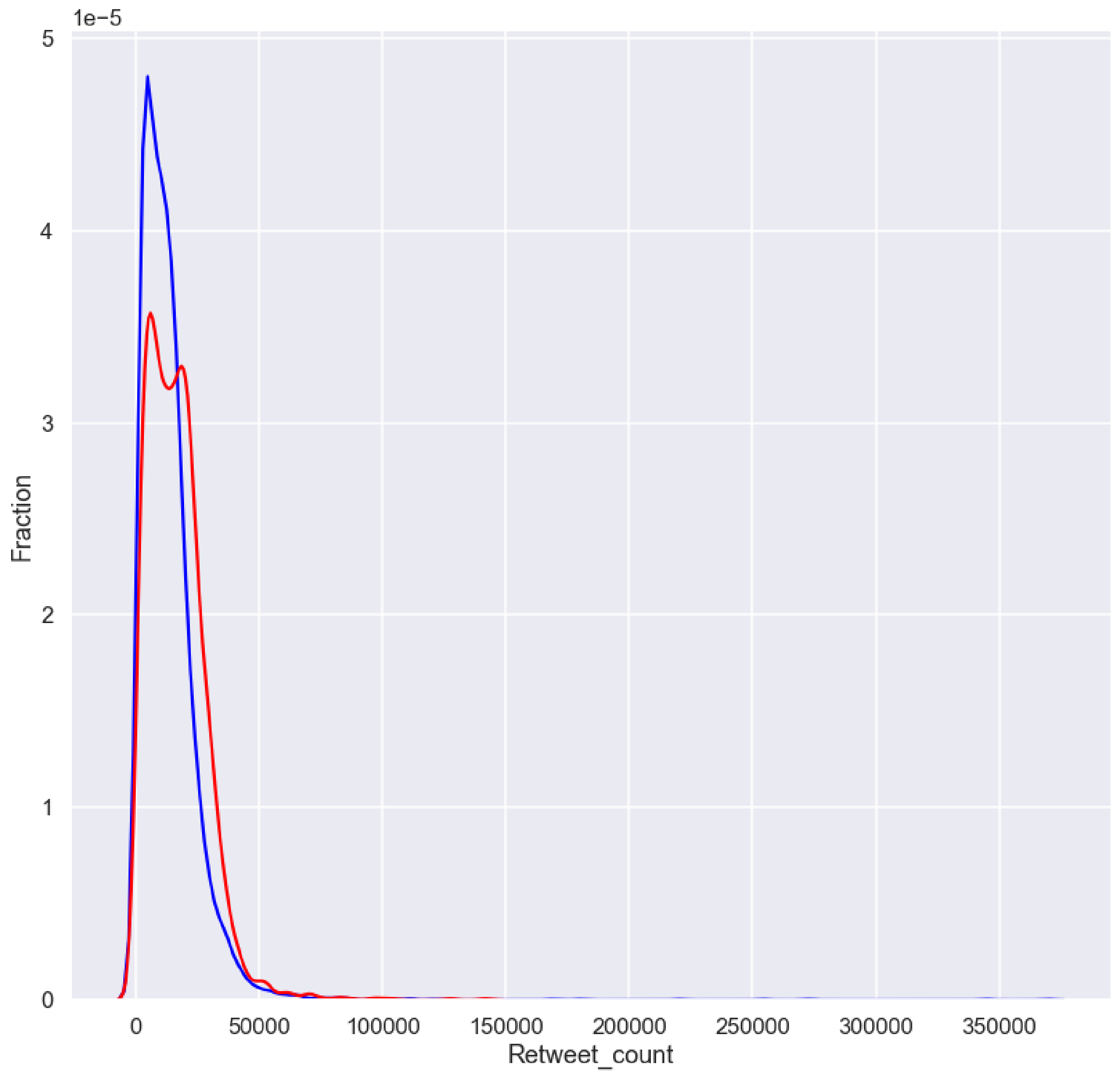
In [49]:

```
#### BEGIN SOLUTION

#### END SOLUTION
```

```
/Users/dd/miniconda3/lib/python3.8/site-packages/seaborn/distributio
ns.py:2551: FutureWarning: `distplot` is a deprecated function and w
ill be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility) or
`kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
/Users/dd/miniconda3/lib/python3.8/site-packages/seaborn/distributio
ns.py:2551: FutureWarning: `distplot` is a deprecated function and w
ill be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility) or
`kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```

Out[49]:

```
Text(0, 0.5, 'Fraction')
```



Discussion: Does sentiment affect retweet count?

BEGIN SOLUTION

END SOLUTION

PART 8 - Analyze Up-to-Date Data (Optional)

In this part, you can analyze the data "10-24-2020.json" that contains the tweets up to October 24th 2020. Notice that some of the columns have different names, but the data contents are mostly the same. You can follow similar process as before. Ths part is only for self interest, and won't be graded. Feel free to add additional cells below to achieve the task. Have fun!

Find Something interesting (Optional for Individuals)

Is there still something interesting to find in this data set? Use your own imagination to ask some good questions. Don't be bias and look for the answer in data. Don't ask us what we want, because we do not know either. This will be for EXTRA CREDIT for individuals but part of the regular assignment for groups. Add any cells below.

Submission Instructions

File Name: Please name the file as __midsemester.jpynb
Submit To: Canvas → Assignments → midsemester
Warning: Failure to follow directions may result in loss points.

Created by Andy Guna @2018 Credits: Josh Hug, and Berkeley Data Science Group, Steve Skiena, David Rodreguez