Homework 2: Pandas, Regular Expressions, Visualizations

Due Date: Mon 4/15, 11:59 pm KST

Collaboration Policy: You may talk with others about the homework, but we ask that you **write your solutions individually**. If you do discuss the assignments with others, please **include their names** in the following line.

Collaborators: *list collaborators here (if applicable)*

Score Breakdown

Question	Points
Question 1a	2
Question 1b	1
Question 1c	2
Question 2	2
Question 3	1
Question 4	2
Question 5a	1
Question 5b	2
Question 5c	2
Question 6a	1
Question 6b	1
Question 6c	1
Question 6d	2
Question 6e	2

Question	Points
Total	22

Initialize your environment

This cell should run without error.

```
In [1]:
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import json
import zipfile
from pprint import pprint # to get a more easily-readable view.

# Ensure that Pandas shows at least 280 characters in columns, so we can see full 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
```

Some common utilities.

```
In [2]:
def utils_head(filename, lines=5):
    """
    Returns the first few lines of a file.

    filename: the name of the file to open
    lines: the number of lines to include

    return: A list of the first few lines from the file.
    """
    from itertools import islice
    with open(filename, "r") as f:
        return list(islice(f, lines))
```

Part 1: Bike Sharing

The data we are exploring is collected from a bike sharing system in Washington D.C.

The variables in this data frame are defined as:

Variable	Description
instant	record index
dteday	date
season	1. spring 2. summer 3. fall 4. winter
yr	year (0: 2011, 1:2012)
mnth	month (1 to 12)
hr	hour (0 to 23)
holiday	whether day is holiday or not
weekday	day of the week
workingday	if day is neither weekend nor holiday
weathersit	1. clear or partly cloudy 2. mist and clouds 3. light snow or rain 4. heavy rain or snow
temp	normalized temperature in Celsius (divided by 41)
atemp	normalized "feels-like" temperature in Celsius (divided by 50)
hum	normalized percent humidity (divided by 100)
windspeed	normalized wind speed (divided by 67)
casual	count of casual users

Variable	Description
registered	count of registered users
cnt	count of total rental bikes including casual and registered

Mount your Google Drive

When you run a code cell, Colab executes it on a temporary cloud instance. Every time you open the notebook, you will be assigned a different machine. All compute state and files saved on the previous machine will be lost. Therefore, you may need to re-download datasets or rerun code after a reset. Here, you can mount your Google drive to the temporary cloud instance's local filesystem using the following code snippet and save files under the specified directory (note that you will have to provide permission every time you run this).

In [3]:
```python
# mount Google drive
from google.colab import drive
drive.mount('/content/drive')

# now you can see files
!echo -e "\nNumber of Google drive files in /content/drive/My Drive/:"
!ls -l "/content/drive/My Drive/" | wc -l
# by the way, you can run any linux command by putting a ! at the start of the line

# by default everything gets executed and saved in /content/
!echo -e "\nCurrent directory:"
!pwd
```

```
Mounted at /content/drive

Number of Google drive files in /content/drive/My Drive/:
9

Current directory:
/content
```

In [4]:
```python
workspace_path = '/content/drive/MyDrive/Lectures/COSE471/hw/hw2/'   # Change this path!
for line in utils_head(workspace_path+'bikeshare.txt'):
    print(line, end="")
```

```
instant,dteday,season,yr,mnth,hr,holiday,weekday,workingday,weathersit,temp,atemp,hum,windspeed,casual,registered,cnt
1,2011-01-01,1,0,1,0,0,6,0,1,0.24,0.2879,0.81,0,3,13,16
2,2011-01-01,1,0,1,1,0,6,0,1,0.22,0.2727,0.8,0,8,32,40
3,2011-01-01,1,0,1,2,0,6,0,1,0.22,0.2727,0.8,0,5,27,32
4,2011-01-01,1,0,1,3,0,6,0,1,0.24,0.2879,0.75,0,3,10,13
```

Loading the data

The following code loads the data into a Pandas DataFrame.

In [5]:
```
bike = pd.read_csv(workspace_path+'bikeshare.txt')
bike.head()
```

Out[5]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0.0	3	13	16
1	2	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0	8	32	40
2	3	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0	5	27	32
3	4	2011-01-01	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0.0	3	10	13
4	5	2011-01-01	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	0.0	0	1	1

Below, we show the shape of the file. You should see that the size of the DataFrame matches the number of lines in the file, minus the header row.

In [6]:
```
bike.shape
```

Out[6]:
```
(17379, 17)
```

Question 1: Data Preparation

A few of the variables that are numeric/integer actually encode categorical data. These include `holiday`, `weekday`, `workingday`, and `weathersit`. In the following problem, we will convert these four variables to strings specifying the categories. In particular, use 3-letter labels (`Sun`, `Mon`, `Tue`, `Wed`, `Thu`, `Fri`, and `Sat`) for `weekday`. You may simply use `yes`/`no` for `holiday` and `workingday`.

In this exercise we will *mutate* the data frame, **overwriting the corresponding variables in the data frame.** However, our notebook will effectively document this in-place data transformation for future readers. Make sure to leave the underlying datafile `bikeshare.txt` unmodified.

Question 1a

Decode the `holiday`, `weekday`, `workingday`, and `weathersit` fields:

1. holiday: Convert to `yes` and `no`. **Hint**: There are fewer holidays...
2. weekday: It turns out that Monday is the day with the most holidays. Mutate the `'weekday'` column to use the 3-letter label (`'Sun'`, `'Mon'`, `'Tue'`, `'Wed'`, `'Thu'`, `'Fri'`, and `'Sat'`) instead of its current numerical values. Note `0` corresponds to `Sun`, `1` to `Mon` and so on.
3. workingday: Convert to `yes` and `no`.
4. weathersit: You should replace each value with one of `Clear`, `Mist`, `Light`, or `Heavy`.

Note: If you want to revert changes, run the cell that reloads the csv.

Hint: One simple approach is to use the [replace](#) method of the pandas DataFrame class. We haven't discussed how to do this so you'll need to look at the documentation. The most concise way is with the approach described in the documentation as `nested-dictonaries`, though there are many possible solutions. E.g. for a DataFrame nested dictionaries, e.g., `{'a': {'b': np.nan}}`, are read as follows: look in column `a` for the value `b` and replace it with `NaN`.

```python
In [7]:  # BEGIN YOUR CODE
         # -----------------------
         factor_dict = {
             'holiday': {0: 'no', 1: 'yes'},
             'weekday': {0: 'Sun', 1: 'Mon', 2: 'Tue', 3: 'Wed', 4: 'Thu', 5: 'Fri', 6: 'Sat'},
             'workingday': {0: 'no', 1: 'yes'},
             'weathersit': {1: 'clear', 2: 'Mist', 3: 'Light', 4: 'Heavy'}

         }
         # -----------------------
         # END YOUR CODE
         bike.replace(factor_dict, inplace=True)
         bike.head()
```

Out[7]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	no	Sat	no	clear	0.24	0.2879	0.81	0.0	3	13	16
1	2	2011-01-01	1	0	1	1	no	Sat	no	clear	0.22	0.2727	0.80	0.0	8	32	40
2	3	2011-01-01	1	0	1	2	no	Sat	no	clear	0.22	0.2727	0.80	0.0	5	27	32
3	4	2011-01-01	1	0	1	3	no	Sat	no	clear	0.24	0.2879	0.75	0.0	3	10	13
4	5	2011-01-01	1	0	1	4	no	Sat	no	clear	0.24	0.2879	0.75	0.0	0	1	1

In [8]:
```python
assert isinstance(bike, pd.DataFrame) == True
assert bike['holiday'].dtype == np.dtype('O')
assert list(bike['holiday'].iloc[370:375]) == ['no', 'no', 'yes', 'yes', 'yes']
assert bike['weekday'].dtype == np.dtype('O')
assert bike['workingday'].dtype == np.dtype('O')
assert bike['weathersit'].dtype == np.dtype('O')
assert bike.shape == (17379, 17) or bike.shape == (17379, 18)
assert list(bike['weekday'].iloc[::2000]) == ['Sat', 'Tue', 'Mon', 'Mon', 'Mon', 'Sun', 'Sun', 'Sat', 'Sun']

print('Passed all unit tests!')
```

Passed all unit tests!

Question 1b

How many entries in the data correspond to holidays? Set the variable `num_holidays` to this value.

Hint: `value_counts`

In [9]:
```python
num_holidays = bike['holiday'].value_counts()['yes']
```

In [10]:
```python
assert num_holidays == 500
assert 1 <= num_holidays <= 10000

print('Passed all unit tests!')
```

Passed all unit tests!

Question 1c (Computing Daily Total Counts)

The granularity of this data is at the hourly level. However, for some of the analysis we will also want to compute daily statistics. In particular, in the next few questions we will be analyzing the daily number of registered and unregistered users.

Construct a data frame named `daily_counts` indexed by `dteday` with the following columns:

- `casual` : total number of casual riders for each day
- `registered` : total number of registered riders for each day
- `workingday` : whether that day is a working day or not (`yes` or `no`)

Hint: `groupby` and `agg` . For the `agg` method, please check the documentation for examples on applying different aggregations per column. If you use the capability to do different aggregations by column, you can do this task with a single call to `groupby` and `agg` . For the `workingday` column we can take any of the values since we are grouping by the day, thus the value will be the same within each group. Take a look at the `'first'` or `'last'` aggregation functions.

```python
In [11]:  # BEGIN YOUR CODE
          # ----------------------
          daily_counts = bike.groupby("dteday").agg(
              casual = pd.NamedAgg(column = "casual", aggfunc = "sum"),
              registered = pd.NamedAgg(column = "registered", aggfunc = "sum"),
              workingday = pd.NamedAgg(column = "workingday", aggfunc = lambda x : x.iloc[0])
          )
          # ----------------------
          # END YOUR CODE
          daily_counts.head()
```

Out[11]:

	casual	registered	workingday
dteday			
2011-01-01	331	654	no
2011-01-02	131	670	no
2011-01-03	120	1229	yes
2011-01-04	108	1454	yes
2011-01-05	82	1518	yes

```
In [12]: assert np.round(daily_counts['casual'].mean()) == 848.0
         assert np.round(daily_counts['casual'].var()) == 471450.0
         assert np.round(daily_counts['registered'].mean()) == 3656.0
         assert np.round(daily_counts['registered'].var()) == 2434400.0
         assert sorted(list(daily_counts['workingday'].value_counts())) == [231, 500]

         print('Passed all unit tests!')
```

```
Passed all unit tests!
```

Part 2: Trump and Tweets

In this part, we will work with Twitter data in order to analyze Donald Trump's tweets.

Let's load data into our notebook. Run the cell below to read tweets from the json file into a list named `all_tweets`.

```
In [13]: with open(workspace_path+"hw2-realdonaldtrump_tweets.json", "r") as f:
             all_tweets = json.load(f)
```

Here is what a typical tweet from `all_tweets` looks like:

```
In [14]: pprint(all_tweets[-1])
```

```
{'contributors': None,
 'coordinates': None,
 'created_at': 'Tue Oct 16 18:40:18 +0000 2018',
 'display_text_range': [0, 174],
 'entities': {'hashtags': [], 'symbols': [], 'urls': [], 'user_mentions': []},
 'favorite_count': 52115,
 'favorited': False,
 'full_text': 'Just spoke with the Crown Prince of Saudi Arabia who totally '
              'denied any knowledge of what took place in their Turkish '
              'Consulate. He was with Secretary of State Mike Pompeo...',
 'geo': None,
 'id': 1052268011900555265,
 'id_str': '1052268011900555265',
 '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': 13493,
 'retweeted': False,
 'source': '<a href="http://twitter.com/download/iphone" '
           'rel="nofollow">Twitter for iPhone</a>',
 'truncated': False,
 'user': {'contributors_enabled': False,
          'created_at': 'Wed Mar 18 13:46:38 +0000 2009',
          'default_profile': False,
          'default_profile_image': False,
          'description': '45th President of the United States of Americaus',
          'entities': {'description': {'urls': []},
                       'url': {'urls': [{'display_url': 'Instagram.com/realDonaldTrump',
                                         'expanded_url': 'http://www.Instagram.com/realDonaldTrump',
                                         'indices': [0, 23],
                                         'url': 'https://t.co/OMxB0x7xC5'}]}},
          'favourites_count': 7,
          'follow_request_sent': False,
          'followers_count': 58311576,
          'following': True,
          'friends_count': 45,
          'geo_enabled': True,
          'has_extended_profile': False,
          'id': 25073877,
```

```
                    'id_str': '25073877',
                    'is_translation_enabled': True,
                    'is_translator': False,
                    'lang': 'en',
                    'listed_count': 100264,
                    'location': 'Washington, DC',
                    'name': 'Donald J. Trump',
                    'notifications': False,
                    'profile_background_color': '6D5C18',
                    'profile_background_image_url': 'http://abs.twimg.com/images/themes/theme1/bg.png',
                    'profile_background_image_url_https': 'https://abs.twimg.com/images/themes/theme1/bg.png',
                    'profile_background_tile': True,
                    'profile_banner_url': 'https://pbs.twimg.com/profile_banners/25073877/1550087458',
                    'profile_image_url': 'http://pbs.twimg.com/profile_images/874276197357596672/kUuht00m_normal.jpg',
                    'profile_image_url_https': 'https://pbs.twimg.com/profile_images/874276197357596672/kUuht00m_normal.jpg',
                    'profile_link_color': '1B95E0',
                    'profile_sidebar_border_color': 'BDDCAD',
                    'profile_sidebar_fill_color': 'C5CEC0',
                    'profile_text_color': '333333',
                    'profile_use_background_image': True,
                    'protected': False,
                    'screen_name': 'realDonaldTrump',
                    'statuses_count': 40563,
                    'time_zone': None,
                    'translator_type': 'regular',
                    'url': 'https://t.co/OMxB0x7xC5',
                    'utc_offset': None,
                    'verified': True}}
```

Question 2

Construct a DataFrame called `trump` containing data from 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.*

```python
In [15]:   # BEGIN YOUR CODE
           # ----------------------
           trump = pd.DataFrame(all_tweets)
           trump = trump.sort_values('id').set_index('id')
           trump['time'] = pd.to_datetime(trump['created_at'])
           trump['text'] = trump['text'].combine_first(trump['full_text'])
           col = ['time', 'source', 'text', 'retweet_count']
           trump = trump[col]
           # ----------------------
           # END YOUR CODE
           trump.head()
```

<ipython-input-15-db505afe4ca2>:5: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  trump['time'] = pd.to_datetime(trump['created_at'])

Out[15]:

id	time	source	text	retweet_count
690171032150237184	2016-01-21 13:56:11+00:00	\Twitter for Android\	"@bigop1: @realDonaldTrump @SarahPalinUSA https://t.co/3kYQGqeVyD"	1059
690171403388104704	2016-01-21 13:57:39+00:00	\Twitter for Android\	"@AmericanAsPie: @glennbeck @SarahPalinUSA Remember when Glenn gave out gifts to ILLEGAL ALIENS at crossing the border? Me too!"	1339
690173226341691392	2016-01-21 14:04:54+00:00	\Twitter for Android\	So sad that @CNN and many others refused to show the massive crowd at the arena yesterday in Oklahoma. Dishonest reporting!	2006
690176882055114758	2016-01-21 14:19:26+00:00	\Twitter for Android\	Sad sack @JebBush has just done another ad on me, with special interest money, saying I won't beat Hillary - I WILL. But he can't beat me.	2266
690180284189310976	2016-01-21 14:32:57+00:00	\Twitter for Android\	Low energy candidate @JebBush has wasted $80 million on his failed presidential campaign. Millions spent on me. He should go home and relax!	2886

```
In [16]:  assert isinstance(trump, pd.DataFrame)
          assert 10000 < trump.shape[0] < 11000
          assert trump.shape[1] >= 4
          assert 831846101179314177 in trump.index
          assert all(col in trump.columns for col in ['time', 'source', 'text', 'retweet_count'])
          assert np.sometrue([('Twitter for iPhone' in s) for s in trump['source'].unique()])
          assert trump['text'].dtype == np.dtype('O')
          assert trump['retweet_count'].dtype == np.dtype('int64')
          assert 753063644578144260 in trump.index

          print('Passed all unit tests!')
```

```
Passed all unit tests!
```

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 [17]:  trump['source'].unique()
```

```
Out[17]:  array(['<a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a>',
                 '<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>',
                 '<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>',
                 '<a href="https://mobile.twitter.com" rel="nofollow">Mobile Web (M5)</a>',
                 '<a href="http://instagram.com" rel="nofollow">Instagram</a>',
                 '<a href="http://twitter.com/#!/download/ipad" rel="nofollow">Twitter for iPad</a>',
                 '<a href="https://studio.twitter.com" rel="nofollow">Media Studio</a>',
                 '<a href="https://periscope.tv" rel="nofollow">Periscope</a>',
                 '<a href="https://ads.twitter.com" rel="nofollow">Twitter Ads</a>',
                 '<a href="https://studio.twitter.com" rel="nofollow">Twitter Media Studio</a>'],
                dtype=object)
```
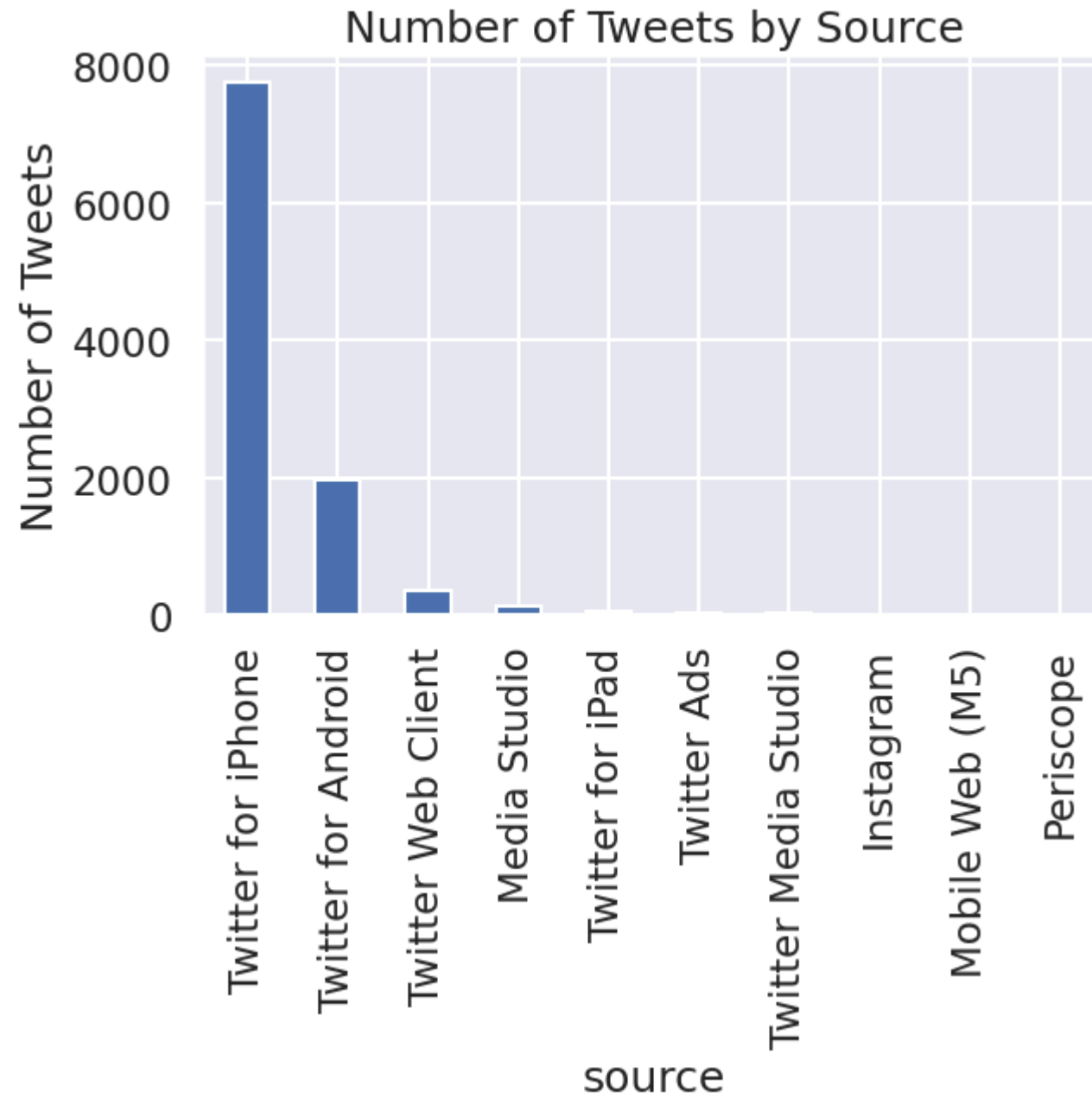
Question 3

Notice how sources like "Twitter for Android" or "Instagram" are surrounded by HTML tags. In the cell below, clean up the `source` field by removing the HTML tags from each `source` entry.

Hints:

- Use `trump['source'].str.replace` along with a regular expression.

- You may find it helpful to experiment with regular expressions at regex101.com.

In [18]:
```python
# BEGIN YOUR CODE
# ----------------------
pattern = r'<[^>}]+>'
trump['source'] = trump['source'].str.replace(pattern, '', regex = True)
# ----------------------
# END YOUR CODE
```

In [19]:
```python
assert set(trump['source'].unique()) == set(['Twitter for Android', 'Twitter for iPhone', 'Twitter Web Client',
        'Mobile Web (M5)', 'Instagram', 'Twitter for iPad', 'Media Studio',
        'Periscope', 'Twitter Ads', 'Twitter Media Studio'])

print('Passed all unit tests!')
```

```
Passed all unit tests!
```

In the following plot, we see that there are two device types that are more commonly used than others.

In [20]:
```python
plt.figure(figsize=(6, 4))
trump['source'].value_counts().plot(kind="bar")
plt.ylabel("Number of Tweets")
plt.title("Number of Tweets by Source");
```

Number of Tweets by Source

Question 4

Now that we have cleaned up the `source` field, let's now look at which device Trump 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)

```python
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

trump['year'] = trump['time'].apply(year_fraction)
```

Now, use `sns.distplot` to overlay the distributions of Trump's 2 most frequently used web technologies over the years.

```python
# BEGIN YOUR CODE
# ----------------------
top_devices = trump['source'].value_counts().nlargest(2).index.tolist()
for device in top_devices:
  sns.distplot(trump[['year']][trump['source'] == device], label = device.split()[-1])
plt.title("Distributions of Tweet Sources Over Years")
plt.legend(loc = 'upper right')
# ----------------------
# END YOUR CODE
```

```
<ipython-input-22-7a0bcd42a8ad>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(trump[['year']][trump['source'] == device], label = device.split()[-1])
<ipython-input-22-7a0bcd42a8ad>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(trump[['year']][trump['source'] == device], label = device.split()[-1])
```
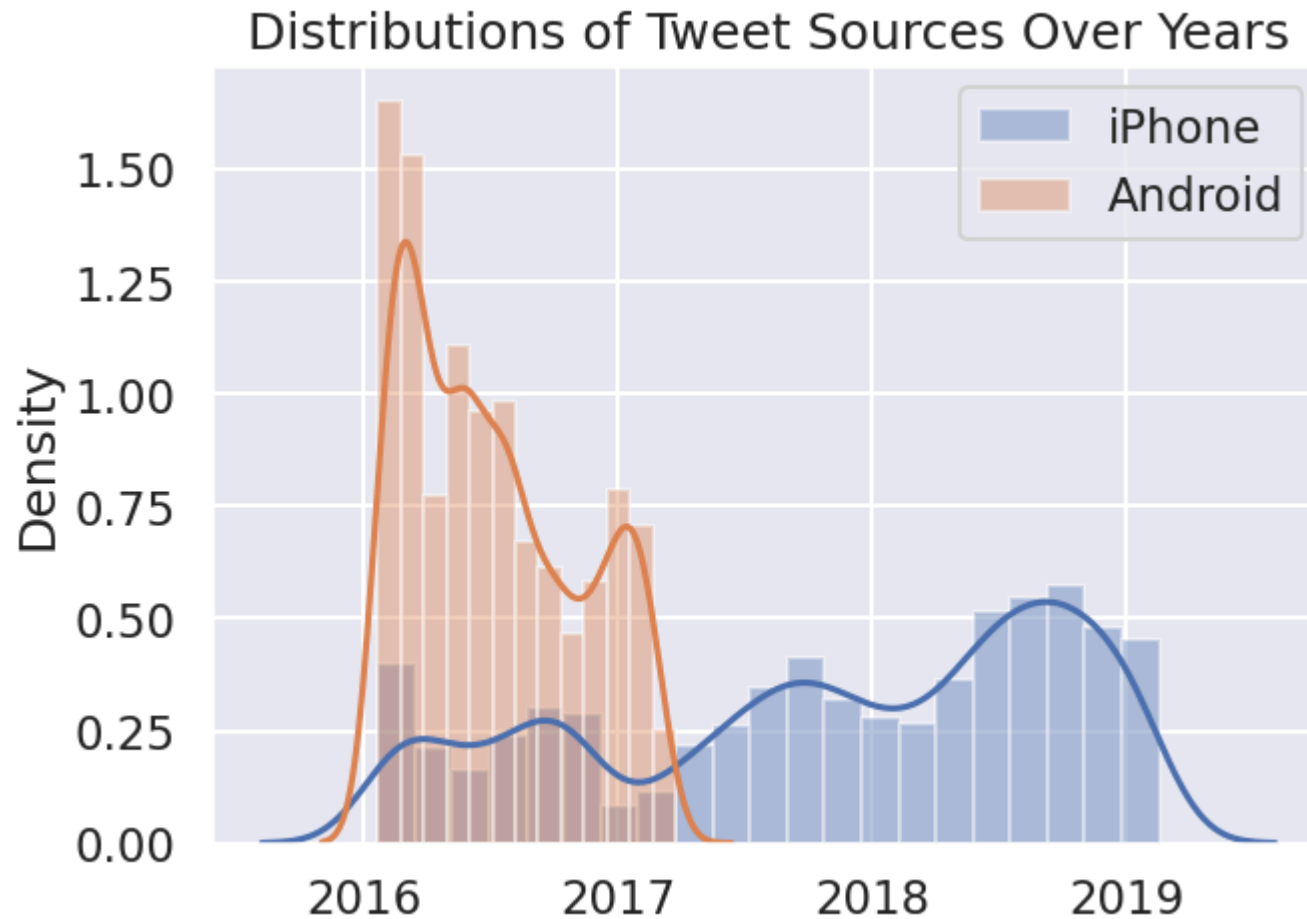
Out[22]:   <matplotlib.legend.Legend at 0x7925301f2710>

Distributions of Tweet Sources Over Years

Question 5

Is there a difference between Trump's 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 device come at different times than his tweets from an iPhone. Note that Twitter gives us his tweets in the UTC timezone (notice the `+0000` in the first few tweets).

```
In [23]:  for tweet in all_tweets[:3]:
              print(tweet['created_at'])
```

```
Wed Oct 12 14:00:48 +0000 2016
Wed Oct 12 13:46:43 +0000 2016
Wed Oct 12 12:59:05 +0000 2016
```

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 [24]:
```python
trump['est_time'] = (
    trump['time'].dt.tz_convert("UTC") # Set initial timezone to UTC
                 .dt.tz_convert("EST") # Convert to Eastern Time
)
trump.head()
```

Out[24]:

id	time	source	text	retweet_count	year	est_time
690171032150237184	2016-01-21 13:56:11+00:00	Twitter for Android	"@bigop1: @realDonaldTrump @SarahPalinUSA https://t.co/3kYQGqeVyD"	1059	2016.054645	2016-01-21 08:56:11-05:00
690171403388104704	2016-01-21 13:57:39+00:00	Twitter for Android	"@AmericanAsPie: @glennbeck @SarahPalinUSA Remember when Glenn gave out gifts to ILLEGAL ALIENS at crossing the border? Me too!"	1339	2016.054645	2016-01-21 08:57:39-05:00
690173226341691392	2016-01-21 14:04:54+00:00	Twitter for Android	So sad that @CNN and many others refused to show the massive crowd at the arena yesterday in Oklahoma. Dishonest reporting!	2006	2016.054645	2016-01-21 09:04:54-05:00
690176882055114758	2016-01-21 14:19:26+00:00	Twitter for Android	Sad sack @JebBush has just done another ad on me, with special interest money, saying I won't beat Hillary - I WILL. But he can't beat me.	2266	2016.054645	2016-01-21 09:19:26-05:00
690180284189310976	2016-01-21 14:32:57+00:00	Twitter for Android	Low energy candidate @JebBush has wasted $80 million on his failed presidential campaign. Millions spent on me. He should go home and relax!	2886	2016.054645	2016-01-21 09:32:57-05:00

Question 5a

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

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

- **Hint:** See the cell above for an example of working with dt accessors.

In [25]:
```python
# BEGIN YOUR CODE
# ----------------------
trump['hour'] = trump['est_time'].dt.hour + trump['est_time'].dt.minute / 60 + trump['est_time'].dt.second / 3600
# ----------------------
# END YOUR CODE
```

In [26]:
```python
assert np.isclose(trump.loc[690171032150237184]['hour'], 8.93639) == True

print('Passed all unit tests!')
```

```
Passed all unit tests!
```

Question 5b

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 [27]:
```python
# BEGIN YOUR CODE
# ----------------------
top_devices = trump['source'].value_counts().nlargest(2).index.tolist()
for device in top_devices:
  sns.distplot(trump[['hour']][trump['source'] == device], label = device.split()[-1], hist = False)
plt.title("Distributions of Tweet Hours for Different Tweet Sources")
plt.legend(loc = 'upper right')
plt.ylabel('fraction')
plt.xlabel('hour')
plt.xticks(range(0, 30, 10))
# ----------------------
# END YOUR CODE
```

```
<ipython-input-27-98632383035e>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(trump[['hour']][trump['source'] == device], label = device.split()[-1], hist = False)
<ipython-input-27-98632383035e>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(trump[['hour']][trump['source'] == device], label = device.split()[-1], hist = False)
```
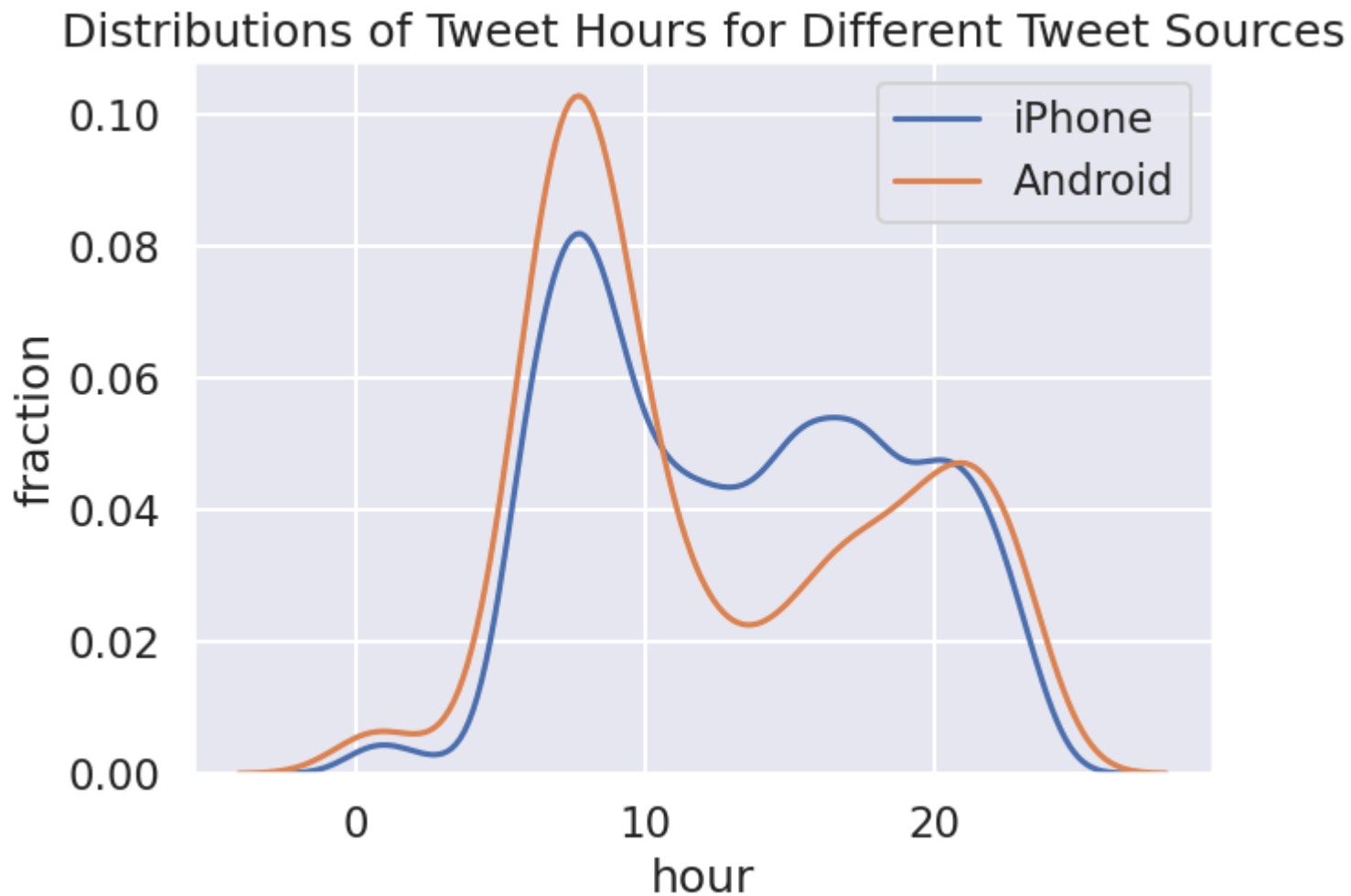
Out[27]:
```
([<matplotlib.axis.XTick at 0x7925302c1300>,
  <matplotlib.axis.XTick at 0x7925302c12d0>,
  <matplotlib.axis.XTick at 0x7925302c2ef0>],
 [Text(0, 0, '0'), Text(10, 0, '10'), Text(20, 0, '20')])
```

Distributions of Tweet Hours for Different Tweet Sources

Question 5c

According to this Verge article, Donald Trump switched from an Android to an iPhone sometime in March 2017.

Let's see if this information significantly changes our plot. Create a figure similar to your figure from question 5b, but this time, only use tweets that were tweeted before 2017.

```
In [28]:    # BEGIN YOUR CODE
            # ---------------------
```

```python
top_devices = trump['source'].value_counts().nlargest(2).index.tolist()
for device in top_devices:
  sns.distplot(trump[['hour']][(trump['source'] == device) & (trump['year'] < 2017)], label = device.split()[-1], hist = False)
plt.title("Distributions of Tweet Hours for Different Tweet Sources (pre-2017)")
plt.legend(loc = 'upper right')
plt.ylabel('fraction')
plt.xlabel('hour')
plt.xticks(range(0, 30, 10))
# ---------------------
# END YOUR CODE
```

```
<ipython-input-28-ece209d2cc85>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(trump[['hour']][(trump['source'] == device) & (trump['year'] < 2017)], label = device.split()[-1], hist = False)
<ipython-input-28-ece209d2cc85>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(trump[['hour']][(trump['source'] == device) & (trump['year'] < 2017)], label = device.split()[-1], hist = False)
```

```
Out[28]:    ([<matplotlib.axis.XTick at 0x79253036b970>,
              <matplotlib.axis.XTick at 0x79253036b940>,
              <matplotlib.axis.XTick at 0x79253034b8e0>],
             [Text(0, 0, '0'), Text(10, 0, '10'), Text(20, 0, '20')])
```
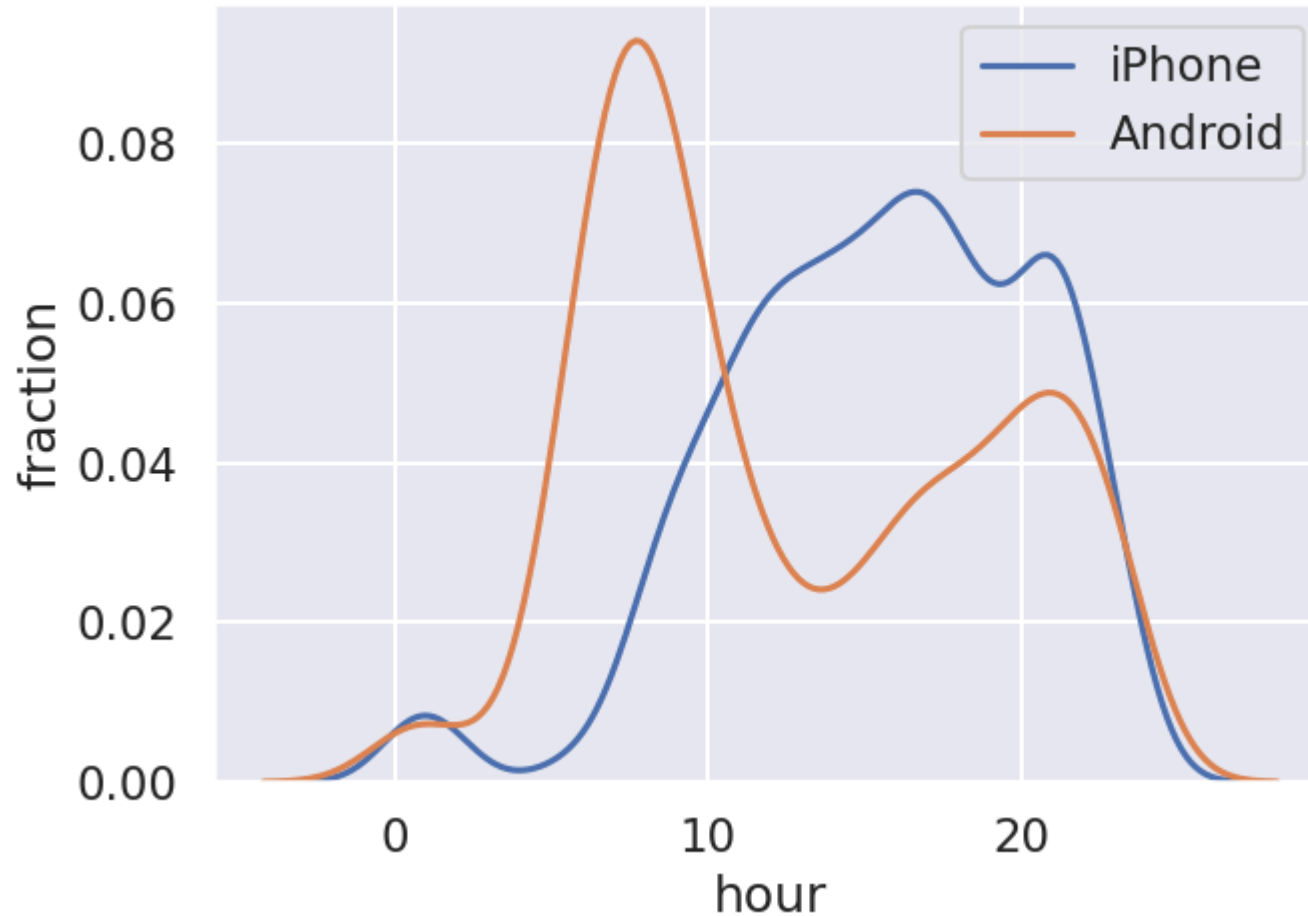
Distributions of Tweet Hours for Different Tweet Sources (pre-2017)

Question 5d

During the campaign, it was theorized that Donald Trump's tweets from Android devices were written by him personally, and the tweets from iPhones were from his staff. Does your figure give support to this theory? What kinds of additional analysis could help support or reject this claim?

Answer: `5c graph represents the usage patterns of source devices before 2017, over time. For Android, it shows a global maximum around hour 8 and a local maximum around hour 21. For iPhone, it indicates a global maximum around`

hour 16 and a local maximum around hour 21. The patterns of usage for both source devices differ over time.
The 5b graph represents the usage patterns of source devices over the entire period. While before 2017, the global
maximum for iPhone occurred around hour 16, over the entire period it occurs around hour 8.
He switched from using an Android phone to an iPhone in March 2017, after which his usage pattern with only iPhone
resembled the pattern when he used Android, indicating that his staff used an iPhone.

Part 3: 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) 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 [29]:  print(''.join(open(workspace_path+"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]
```

Question 6

As you can see, the lexicon contains emojis too! Each row contains a word and the *polarity* of that word, measuring how positive or negative the word is.

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

Question 6a

Read in the lexicon into a DataFrame called `sent`. The index of the DataFrame should be the words in the lexicon. `sent` should have one column named `polarity`, storing the polarity of each word.

- **Hint:** The `pd.read_csv` function may help here.

In [30]:
```python
# BEGIN YOUR CODE
# ------------------------
path = workspace_path + 'vader_lexicon.txt'
col = ['token', 'polarity']
sent = pd.read_csv(path, delimiter = '\t', header = None, usecols = [0, 1])
sent.columns = col
sent = sent.set_index('token')
# ------------------------
# END YOUR CODE
sent.head()
```

Out[30]:

token	polarity
$:	-1.5
%)	-0.4
%-)	-1.5
&-:	-0.4
&:	-0.7

In [31]:
```python
assert np.allclose(sent['polarity'].head(), [-1.5, -0.4, -1.5, -0.4, -0.7]) == True
assert list(sent.index[5000:5005]) == ['paranoids', 'pardon', 'pardoned', 'pardoning', 'pardons']

print('Passed all unit tests!')
```

```
Passed all unit tests!
```

Question 6b

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 `trump` DataFrame to be the lowercased text of each tweet.

In [32]:
```python
# BEGIN YOUR CODE
# -----------------------
trump['text'] = trump['text'].str.lower()
# -----------------------
# END YOUR CODE
trump.head()
```

Out[32]:

id	time	source	text	retweet_count	year	est_time	hour
690171032150237184	2016-01-21 13:56:11+00:00	Twitter for Android	"@bigop1: @realdonaldtrump @sarahpalinusa https://t.co/3kyqgqevyd"	1059	2016.054645	2016-01-21 08:56:11-05:00	8.936389
690171403388104704	2016-01-21 13:57:39+00:00	Twitter for Android	"@americanaspie: @glennbeck @sarahpalinusa remember when glenn gave out gifts to illegal aliens at crossing the border? me too!"	1339	2016.054645	2016-01-21 08:57:39-05:00	8.960833
690173226341691392	2016-01-21 14:04:54+00:00	Twitter for Android	so sad that @cnn and many others refused to show the massive crowd at the arena yesterday in oklahoma. dishonest reporting!	2006	2016.054645	2016-01-21 09:04:54-05:00	9.081667
690176882055114758	2016-01-21 14:19:26+00:00	Twitter for Android	sad sack @jebbush has just done another ad on me, with special interest money, saying i won't beat hillary - i will. but he can't beat me.	2266	2016.054645	2016-01-21 09:19:26-05:00	9.323889
690180284189310976	2016-01-21 14:32:57+00:00	Twitter for Android	low energy candidate @jebbush has wasted $80 million on his failed presidential campaign. millions spent on me. he should go home and relax!	2886	2016.054645	2016-01-21 09:32:57-05:00	9.549167

In [33]:
```python
assert trump['text'].loc[884740553040175104] == 'working hard to get the olympics for the united states (l.a.). stay tuned!'

print('Passed all unit tests!')
```

Passed all unit tests!

Question 6c

Now, let's get rid of punctuation since it will cause us to fail to match words. Create a new column called `no_punc` in the `trump` DataFrame 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 [34]:  # BEGIN YOUR CODE
          # -----------------------
          punct_re = r'[^\w\s]'  # Save your regex in punct_re
          trump['no_punc'] = trump['text'].str.replace(punct_re, ' ', regex = True)
          # -----------------------
          # END YOUR CODE
```

```
In [35]:  assert re.search(punct_re, 'this') == None
          assert re.search(punct_re, 'this is not ok.') != None
          assert re.search(punct_re, 'this#is#ok') != None
          assert re.search(punct_re, 'this^is ok') != None

          print('Passed all unit tests!')
```

Passed all unit tests!

Question 6d

Now, let's convert the tweets into what's called a *tidy format* to make the sentiments easier to calculate. Use the `no_punc` column of `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 your DataFrame may look different from the one above. 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` DataFrame, 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 [36]:
```python
# BEGIN YOUR CODE
# ----------------------
col = ['num', 'word']
tidy_format = pd.DataFrame(trump['no_punc'].str.split(expand = True).stack())
tidy_format = tidy_format.reset_index(level = 1)
tidy_format.columns = col
# ----------------------
# END YOUR CODE
tidy_format.head()
```

Out[36]:

id	num	word
690171032150237184	0	bigop1
690171032150237184	1	realdonaldtrump
690171032150237184	2	sarahpalinusa
690171032150237184	3	https
690171032150237184	4	t

In [37]:
```python
assert tidy_format.loc[894661651760377856].shape == (27,2)
assert ' '.join(list(tidy_format.loc[894661651760377856]['word'])) == 'i think senator blumenthal should take a nice long vacation

print('Passed all unit tests!')
```

Passed all unit tests!

Question 6e

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.

Hints:

- You will need to merge the `tidy_format` and `sent` tables and group the final answer.
- If certain words are not found in the `sent` table, set their polarities to 0.

In [38]:
```python
# BEGIN YOUR CODE
# ----------------------
tmpdict = sent['polarity'].to_dict() # express dictionary key : word , value : polarity
trump['polarity'] = tidy_format['word'].apply(lambda x : tmpdict.get(x, 0)).groupby('id').agg(sum)
# ----------------------
# END YOUR CODE
trump[['text', 'polarity']].head()
```

Out[38]:

id	text	polarity
690171032150237184	"@bigop1: @realdonaldtrump @sarahpalinusa https://t.co/3kyqgqevyd"	0.0
690171403388104704	"@americanaspie: @glennbeck @sarahpalinusa remember when glenn gave out gifts to illegal aliens at crossing the border? me too!"	-2.6
690173226341691392	so sad that @cnn and many others refused to show the massive crowd at the arena yesterday in oklahoma. dishonest reporting!	-6.0
690176882055114758	sad sack @jebbush has just done another ad on me, with special interest money, saying i won't beat hillary - i will. but he can't beat me.	4.3
690180284189310976	low energy candidate @jebbush has wasted $80 million on his failed presidential campaign. millions spent on me. he should go home and relax!	-2.6

```
In [40]: assert np.allclose(trump.loc[744701872456536064, 'polarity'], 8.4)
         assert np.allclose(trump.loc[745304731346702336, 'polarity'], 2.5)
         assert np.allclose(trump.loc[744519497764184064, 'polarity'], 1.7)
         assert np.allclose(trump.loc[894661651760377856, 'polarity'], 0.2)
         assert np.allclose(trump.loc[894620077634592769, 'polarity'], 5.4)

         print('Passed all unit tests!')
```

```
Passed all unit tests!
```

Now we have a measure of the sentiment of each of his tweets! Note that this calculation is rather basic; you can read over the VADER readme to understand a more robust sentiment analysis.

Now, run the cells below to see the most positive and most negative tweets from Trump in your dataset:

```
In [41]: print('Most negative tweets:')
         for t in trump.sort_values('polarity').head()['text']:
             print('\n  ', t)
```

```
Most negative tweets:

   the trump portrait of an unsustainable border crisis is dead on.  "in the last two years, ice officers made 266,000 arrests of a
liens with criminal records, including those charged or convicted of 100,000 assaults, 30,000 sex crimes &amp; 4000 violent killing
s."  america's southern....

   it is outrageous that poisonous synthetic heroin fentanyl comes pouring into the u.s. postal system from china. we can, and mus
t, end this now! the senate should pass the stop act - and firmly stop this poison from killing our children and destroying our cou
ntry. no more delay!

   the rigged russian witch hunt goes on and on as the  "originators and founders"  of this scam continue to be fired and demoted f
or their corrupt and illegal activity. all credibility is gone from this terrible hoax, and much more will be lost as it proceeds.
no collusion!

   ...this evil anti-semitic attack is an assault on humanity. it will take all of us working together to extract the poison of ant
i-semitism from our world. we must unite to conquer hate.

   james comey is a proven leaker &amp; liar. virtually everyone in washington thought he should be fired for the terrible job he d
id-until he was, in fact, fired. he leaked classified information, for which he should be prosecuted. he lied to congress under oat
h. he is a weak and.....
```

```
In [42]: print('Most positive tweets:')
         for t in trump.sort_values('polarity', ascending=False).head()['text']:
             print('\n  ', t)
```

```
Most positive tweets:

    congratulations to patrick reed on his great and courageous masters win! when patrick had his amazing win at doral 5 years ago,
people saw his great talent, and a bright future ahead. now he is the masters champion!

    congratulations to a truly great football team, the clemson tigers, on an incredible win last night against a powerful alabama t
eam. a big win also for the great state of south carolina. look forward to seeing the team, and their brilliant coach, for the seco
nd time at the w.h.

    my supporters are the smartest, strongest, most hard working and most loyal that we have seen in our countries history. it is a
beautiful thing to watch as we win elections and gather support from all over the country. as we get stronger, so does our country.
best numbers ever!

    thank you to all of my great supporters, really big progress being made. other countries wanting to fix crazy trade deals. econo
my is roaring. supreme court pick getting great reviews. new poll says trump, at over 90%, is the most popular republican in histor
y of the party. wow!

    thank you, @wvgovernor jim justice, for that warm introduction. tonight, it was my great honor to attend the "greenbrier classi
c - salute to service dinner" in west virginia! god bless our veterans. god bless america – and happy independence day to all! htt
ps://t.co/v35qvcn8m6
```

Now, let's try looking at the distributions of sentiments for tweets containing certain keywords.

In the cell below, we create a single plot showing both the distribution of tweet sentiments for tweets containing `nytimes` , as well as the distribution of tweet sentiments for tweets containing `fox` . Here, we notice that the president appears to say more positive things about Fox than the New York Times.

In [43]:
```python
sns.distplot(trump[trump['text'].str.lower().str.contains("nytimes")]['polarity'], label = 'nytimes')
sns.distplot(trump[trump['text'].str.lower().str.contains("fox")]['polarity'], label = 'fox')
plt.title('Distributions of Tweet Polarities (nytimes vs. fox)')
plt.legend();
```

```
<ipython-input-43-2c44e2f04d2d>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(trump[trump['text'].str.lower().str.contains("nytimes")]['polarity'], label = 'nytimes')
<ipython-input-43-2c44e2f04d2d>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(trump[trump['text'].str.lower().str.contains("fox")]['polarity'], label = 'fox')
```
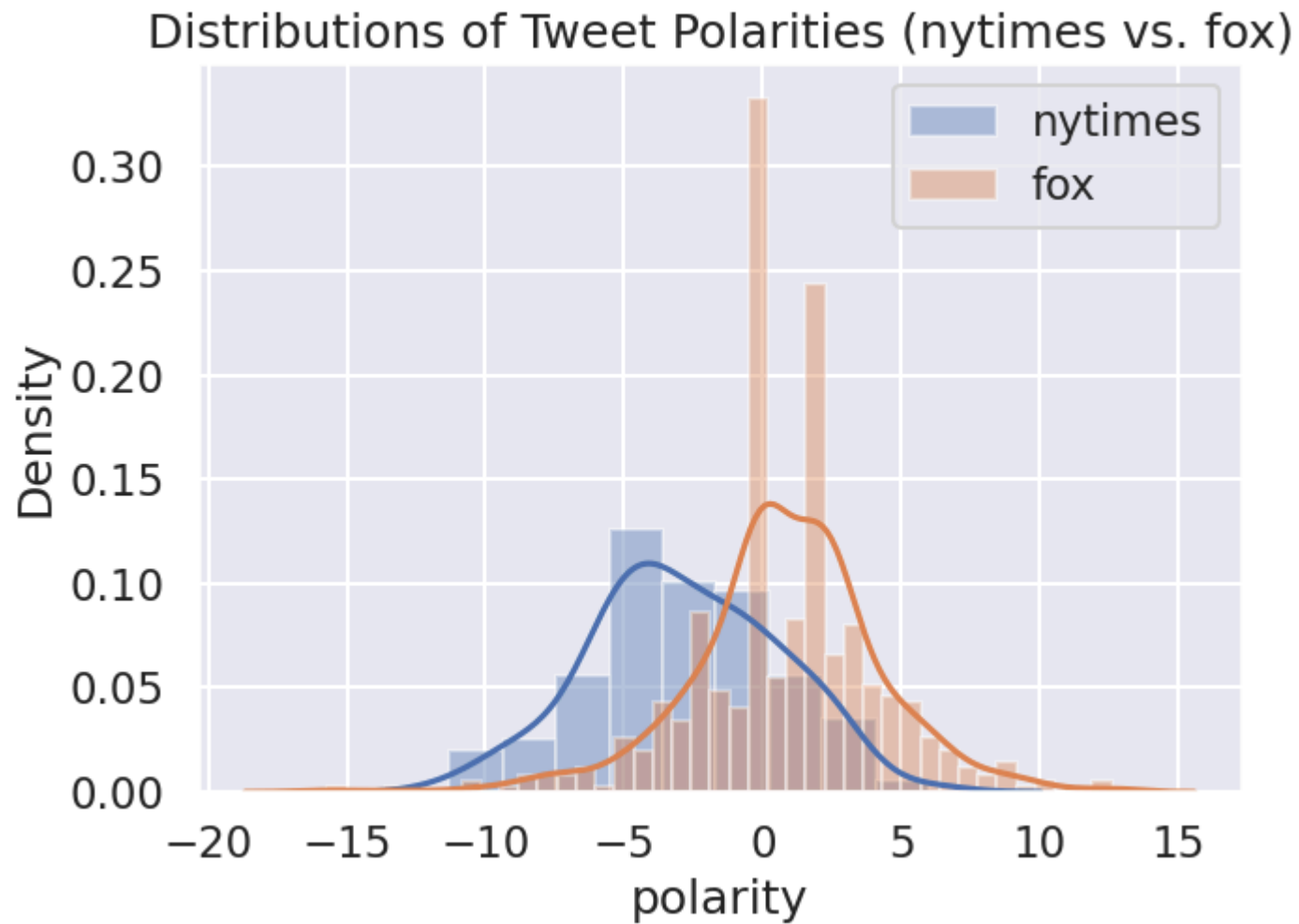
Distributions of Tweet Polarities (nytimes vs. fox)

Congratulations! You have completed HW2.

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output.,

Please generate pdf as follows and submit it to Gradescope.

File > Print Preview > Print > Save as pdf

Please save before submitting!