

```
In [1]: # ! pip install -U okpy # Uncomment if you have an error
        from client.api.notebook import Notebook
        ok = Notebook('hw2.ok')
```

```
=====
Assignment: hw2
OK, version v1.18.1
=====
```

Homework 2: Exploratory Data Analysis (EDA)

Due Date: Fri 4/9, 11:59 pm PST

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
Total	22

Initialize your environment

This cell should run without error.

```
In [2]: 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.
import ds100_utils

# Ensure that Pandas shows at least 280 characters in columns, so we can see full
pd.set_option('max_colwidth', 280)
%matplotlib inline
plt.style.use('fivethirtyeight')
import seaborn as sns
sns.set()
sns.set_context("talk")
import re
print("Library Working")
```

Library Working

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)

Variable	Description
windspeed	normalized wind speed (divided by 67)
casual	count of casual users
registered	count of registered users
cnt	count of total rental bikes including casual and registered

```
In [3]: for line in ds100_utils.head('data/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 [4]: bike = pd.read_csv('data/bikeshare.txt')
        bike.head()
```

```
Out[4]:
```

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp
0	1	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.28
1	2	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.27
2	3	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.27
3	4	2011-01-01	1	0	1	3	0	6	0	1	0.24	0.28
4	5	2011-01-01	1	0	1	4	0	6	0	1	0.24	0.28

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 [5]: bike.shape
```

```
Out[5]: (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-dictionaries`, 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`.

In [6]:

```
# BEGIN YOUR CODE
# -----
bike = bike.replace({"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"}})
# weathersit -> Clear : 1, Mist : 2, Light : 3, Heavy : 4
# -----
# END YOUR CODE
bike.head()
```

Out[6]:

instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	ater
---------	--------	--------	----	------	----	---------	---------	------------	------------	------	------

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	ater
0	1	2011-01-01	1	0	1	0	no	Sat	no	Clear	0.24	0.28
1	2	2011-01-01	1	0	1	1	no	Sat	no	Clear	0.22	0.27
2	3	2011-01-01	1	0	1	2	no	Sat	no	Clear	0.22	0.27
3	4	2011-01-01	1	0	1	3	no	Sat	no	Clear	0.24	0.28
4	5	2011-01-01	1	0	1	4	no	Sat	no	Clear	0.24	0.28

In [7]:

```
ok.grade("q1a");
```

```
~~~~~
Running tests
```

```
-----
Test summary
  Passed: 10
  Failed: 0
[ooooooooook] 100.0% passed
```

Question 1b

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

Hint: `value_counts`

In [8]:

```
num_holidays = bike["holiday"].value_counts()["yes"]
```

In [9]:

```
ok.grade("q1b");
```

```
~~~~~
Running tests
```

```
-----
Test summary
  Passed: 2
  Failed: 0
[ooooooooook] 100.0% passed
```

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.

```
In [10]: # BEGIN YOUR CODE
# -----
daily_counts = bike[["casual", "registered"]].groupby(bike["dteday"]).agg(sum)
work = bike[["workingday"]].groupby(bike["dteday"]).agg(lambda x: x.iloc[0])
result = pd.concat([daily_counts, work], axis=1) # 두 결과 값 합치기. 한 줄의 코드로 정리할
daily_counts = result
# -----
# END YOUR CODE
daily_counts.head()
```

Out[10]:

	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 [11]: ok.grade("q1c");
```

~~~~~  
Running tests

-----  
Test summary

Passed: 5

Failed: 0

[ooooooooook] 100.0% passed

## 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 [12]: with open("data/hw2-realdonaldtrump_tweets.json", "r") as f:
         all_tweets = json.load(f)
```

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

```
In [13]: pprint(all_tweets[1])

{'contributors': None,
 'coordinates': None,
 'created_at': 'Wed Oct 12 13:46:43 +0000 2016',
 'entities': {'hashtags': [], 'symbols': [], 'urls': [], 'user_mentions': []},
 'favorite_count': 54117,
 'favorited': False,
 'geo': None,
 'id': 786201435486781440,
 'id_str': '786201435486781440',
 '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': {'attributes': {},
           'bounding_box': {'coordinates': [[[-179.231086, 13.182335],
                                             [179.859685, 13.182335],
                                             [179.859685, 71.434357],
                                             [-179.231086, 71.434357]]],
                             'type': 'Polygon'},
           'contained_within': [],
           'country': 'United States',
           'country_code': 'US',
           'full_name': 'United States',
           'id': '96683cc9126741d1',
           'name': 'United States',
           'place_type': 'country',
           'url': 'https://api.twitter.com/1.1/geo/id/96683cc9126741d1.json'},
 'retweet_count': 22609,
 'retweeted': False,
 'source': '<a href="http://twitter.com/download/iphone" '
           'rel="nofollow">Twitter for iPhone</a>',
 'text': 'Very little pick-up by the dishonest media of incredible information '
         'provided by WikiLeaks. So dishonest! Rigged system!',
 '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 America🇺🇸',
          'entities': {'description': {'urls': []}},
          'favourites_count': 12,
          'follow_request_sent': False,
          'followers_count': 35307313,
          'following': False,
          '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': 74225,
'location': 'Washington, DC',
'name': 'Donald J. Trump',
'notifications': False,
'profile_background_color': '6D5C18',
'profile_background_image_url': 'http://pbs.twimg.com/profile_backgrou
nd_images/530021613/trump_scotland__43_of_70_cc.jpg',
'profile_background_image_url_https': 'https://pbs.twimg.com/profile_b
ackground_images/530021613/trump_scotland__43_of_70_cc.jpg',
'profile_background_tile': True,
'profile_banner_url': 'https://pbs.twimg.com/profile_banners/25073877/
1501916634',
'profile_image_url': 'http://pbs.twimg.com/profile_images/874276197357
596672/kUuht00m_normal.jpg',
'profile_image_url_https': 'https://pbs.twimg.com/profile_images/87427
6197357596672/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': 35480,
'time_zone': 'Eastern Time (US & Canada)',
'translator_type': 'regular',
'url': None,
'utc_offset': -14400,
'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 as below.**





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

In [14]:

```
# BEGIN YOUR CODE
# -----
dt = {"id":[],
      "time":[],
      "source":[],
      "text":[],
      "retweet_count":[]}
}
for twit in all_tweets:
    dt["id"].append(twit['id'])
    dt["time"].append(twit["created_at"])
    dt["source"].append(twit["source"])
    if "text" in twit:
        dt["text"].append(twit["text"])
    elif "full_text" in twit:
        dt["text"].append(twit["full_text"])
    dt["retweet_count"].append(twit["retweet_count"])
trump = pd.DataFrame(dt, index=dt["id"], columns=["time", "source",
                                                  "text", "retweet_count"])
trump["time"] = pd.to_datetime(trump["time"]).dt.strftime('%Y-%m-%d %H:%M:%S')
trump["time"] = pd.to_datetime(trump["time"]) # Convert dtype into datetime
# -----
# END YOUR CODE
trump.head()
```

Out[14]:

|                           | time                | source                                                                               |                                                                                           |
|---------------------------|---------------------|--------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| <b>786204978629185536</b> | 2016-10-12 14:00:48 | <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>   | PAY TO PLAY POLITICS. 'https                                                              |
| <b>786201435486781440</b> | 2016-10-12 13:46:43 | <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>   | Very little pick-up by the d<br>incredible inform<br>WikiLeaks. So dishones               |
| <b>786189446274248704</b> | 2016-10-12 12:59:05 | <a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a> | Crooked Hillary Clinton<br>the things she will do<br>there for 30 years -                 |
| <b>786054986534969344</b> | 2016-10-12 04:04:47 | <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>   | Thank you Florida- a MC<br>never been seen before<br>seen again. Let<br>https://          |
| <b>786007502639038464</b> | 2016-10-12 00:56:06 | <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>   | Join me Thursday!<br>Ohio!\nWest<br>noon:\nhttps://t.co/jwbZnQ<br>OH this 7:30pm:\nhttps: |

In [15]:

```
ok.grade("q2");
```

~~~~~  
Running tests
~~~~~

```
Test summary
  Passed: 11
  Failed: 0
[ooooooooook] 100.0% passed
```

---

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

First let's examine the source field:

```
In [16]: trump['source'].unique()

Out[16]: array(['<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for
iPhone</a>',
'<a href="http://twitter.com/download/android" rel="nofollow">Twitter for
Android</a>',
'<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>',
'<a href="https://studio.twitter.com" rel="nofollow">Media Studio</a>',
'<a href="http://twitter.com/#!/download/ipad" rel="nofollow">Twitter for
iPad</a>',
'<a href="http://instagram.com" rel="nofollow">Instagram</a>',
'<a href="https://mobile.twitter.com" rel="nofollow">Mobile Web (M5)</a
>',
'<a href="https://ads.twitter.com" rel="nofollow">Twitter Ads</a>',
'<a href="https://periscope.tv" rel="nofollow">Periscope</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](https://regex101.com).

```
In [17]: # BEGIN YOUR CODE
# -----
trump['source'] = trump['source'].str.replace("<[>]+>",
                                             "", regex=True)
trump["source"] = trump['source'] # 원본데이터 변경
# -----
# END YOUR CODE
```

```
In [18]: ok.grade("q3");
```

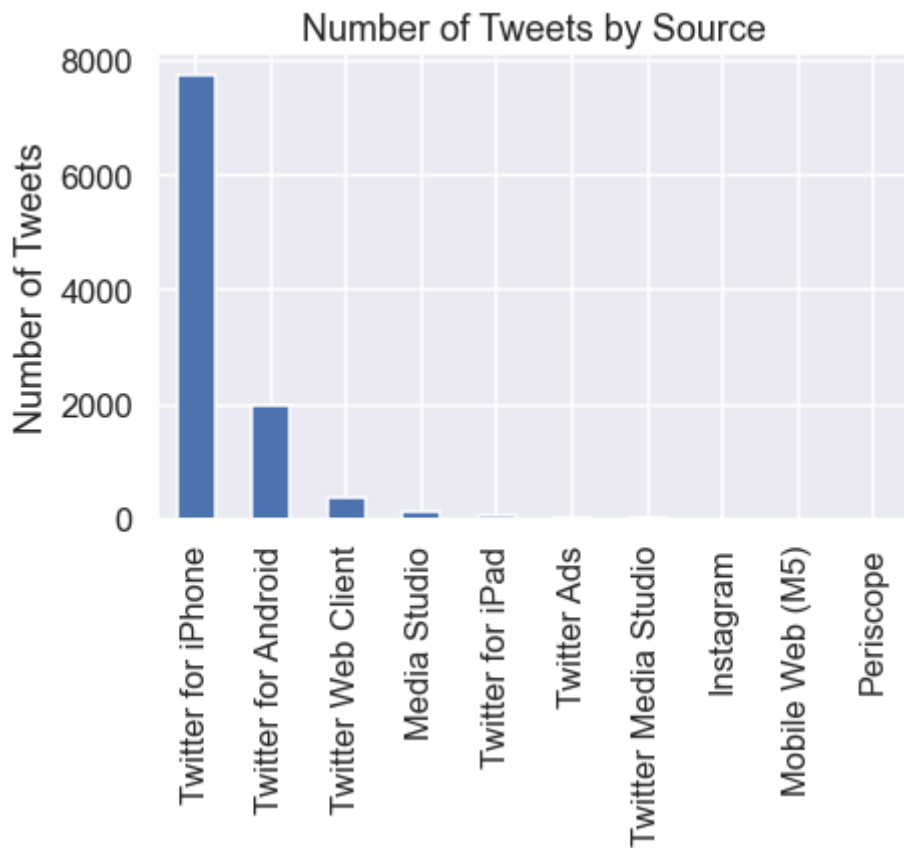
```
~~~~~
Running tests
```

```

Test summary
 Passed: 1
 Failed: 0
[ooooooooook] 100.0% passed
```

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

```
In [19]: plt.figure(figsize=(6, 4))
trump['source'].value_counts().plot(kind="bar")
plt.ylabel("Number of Tweets")
plt.title("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>)

```
In [20]: 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. Your final plot should look like:



```
In [21]: # BEGIN YOUR CODE

sns.distplot(trump[trump['source'].str.contains("iPhone")]['year'], label = 'iPhone')
sns.distplot(trump[trump['source'].str.contains("Android")]['year'], label = 'Android')
plt.title('Distributions of Tweet Sources Over Years')
plt.ylabel('')
plt.legend();

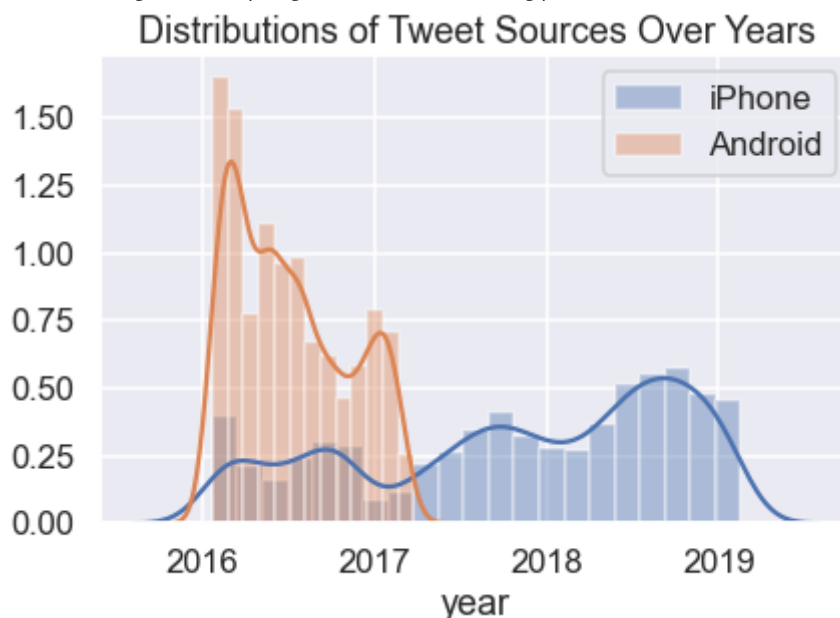
END YOUR CODE
```

/Users/ryan\_kim/Documents/programming/ds/data/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

/Users/ryan\_kim/Documents/programming/ds/data/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)



## 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 [22]:

```
for tweet in all_tweets[:10]:
 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
Wed Oct 12 04:04:47 +0000 2016
Wed Oct 12 00:56:06 +0000 2016
Tue Oct 11 23:04:25 +0000 2016
Tue Oct 11 21:35:41 +0000 2016
Tue Oct 11 20:56:33 +0000 2016
Tue Oct 11 18:43:35 +0000 2016
Tue Oct 11 18:29:59 +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 [23]:

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

Out[23]:

|                           | time                | source              | text                                                                                                                             | retweet_count |
|---------------------------|---------------------|---------------------|----------------------------------------------------------------------------------------------------------------------------------|---------------|
| <b>786204978629185536</b> | 2016-10-12 14:00:48 | Twitter for iPhone  | PAY TO PLAY POLITICS. \n#CrookedHillary <a href="https://t.co/wjsl8ITVvk">https://t.co/wjsl8ITVvk</a>                            | 24915         |
| <b>786201435486781440</b> | 2016-10-12 13:46:43 | Twitter for iPhone  | Very little pick-up by the dishonest media of incredible information provided by WikiLeaks. So dishonest! Rigged system!         | 22609         |
| <b>786189446274248704</b> | 2016-10-12 12:59:05 | Twitter for Android | Crooked Hillary Clinton likes to talk about the things she will do but she has been there for 30 years - why didn't she do them? | 18329         |

|                    | time                | source             | text                                                                                                                                                   | retweet_count |
|--------------------|---------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| 786054986534969344 | 2016-10-12 04:04:47 | Twitter for iPhone | Thank you Florida- a MOVEMENT that has never been seen before and will never be seen again. Lets get out &...<br>https://t.co/t9XM9wFDZI               | 18789         |
| 786007502639038464 | 2016-10-12 00:56:06 | Twitter for iPhone | Join me Thursday in Florida &...<br>Ohio!\nWest Palm Beach, FL at noon:\nhttps://t.co/jwbZnQhXg9\nCincinnati, OH this 7:30pm:\nhttps://t.co/5w2UhalPlx | 7761          |

## 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 [43]:

```
BEGIN YOUR CODE

trump["est_time"] = trump["est_time"] # 변수 할당을 안해주면 기존의 Data Frame에 삽입이 안됨
trump['hour'] = trump["est_time"].dt.hour + (trump["est_time"].dt.minute / 60) + (trump["est_time"].dt.second / 3600)

END YOUR CODE
```

Out[43]:

```
786204978629185536 0.013333
786201435486781440 0.011944
786189446274248704 0.001389
786054986534969344 0.013056
786007502639038464 0.001667
...
1052344365442588673 0.011667
1052340597900627968 0.012222
1052270676982333442 0.015000
1052268013876064256 0.005278
1052268011900555265 0.005000
Name: est_time, Length: 10370, dtype: float64
```

In [25]:

```
ok.grade("q5a");
```

~~~~~  
Running tests

-----  
Test summary

Passed: 1

Failed: 0

[ooooooooook] 100.0% passed

## 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. Your plot should look similar to the following:



In [26]:

```
BEGIN YOUR CODE

sns.distplot(trump[trump['source'].str.contains("iPhone")]['hour'], label = 'iPhone')
sns.distplot(trump[trump['source'].str.contains("Android")]['hour'], label = 'Android')
plt.title('Distributions of Tweet Hours for Different Tweet Sources')
plt.ylabel('fraction')
plt.legend();

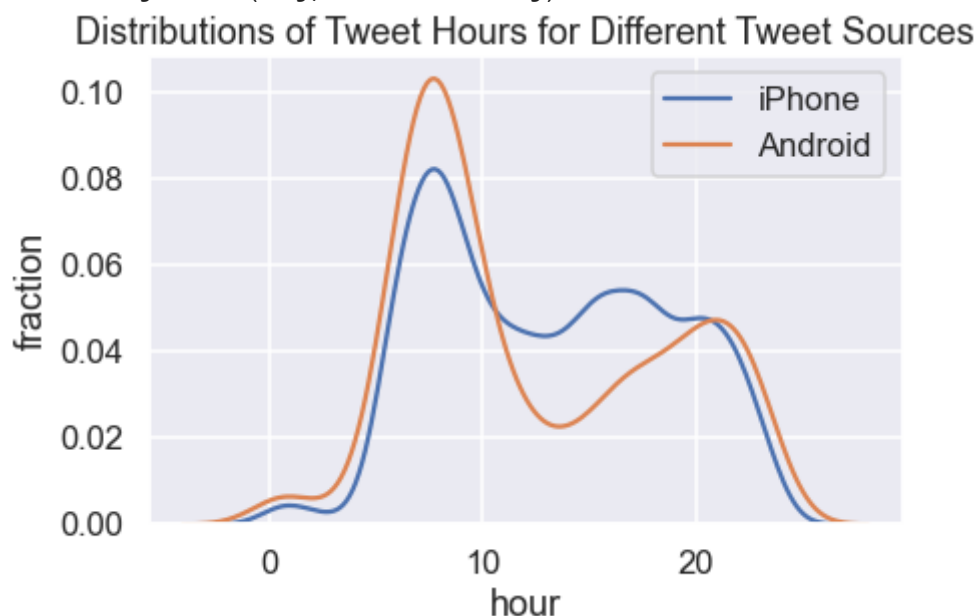
END YOUR CODE
```

/Users/ryan\_kim/Documents/programming/ds/data/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. 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).

warnings.warn(msg, FutureWarning)

/Users/ryan\_kim/Documents/programming/ds/data/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. 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).

warnings.warn(msg, FutureWarning)



## 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. Your plot should look similar to the following:



In [27]:

```
BEGIN YOUR CODE

iphone = trump[trump["est_time"].dt.year <2017][trump['source'].str.contains("iPhone")]
android = trump[trump["est_time"].dt.year <2017][trump['source'].str.contains("Android")]
sns.distplot(iphone["hour"], label = 'iPhone', hist=False)
sns.distplot(android["hour"], label = 'Android', hist=False)
plt.title('Distributions of Tweet Hours for Different Tweet Sources(pre-2017)')
plt.ylabel('fraction')
plt.xticks([0,10,20,30])
plt.legend();

END YOUR CODE
```

<ipython-input-27-7c0af215dda1>:3: UserWarning: Boolean Series key will be reindexed to match DataFrame index.

```
iphone = trump[trump["est_time"].dt.year <2017][trump['source'].str.contains("iPhone")]
```

<ipython-input-27-7c0af215dda1>:4: UserWarning: Boolean Series key will be reindexed to match DataFrame index.

```
android = trump[trump["est_time"].dt.year <2017][trump['source'].str.contains("Android")]
```

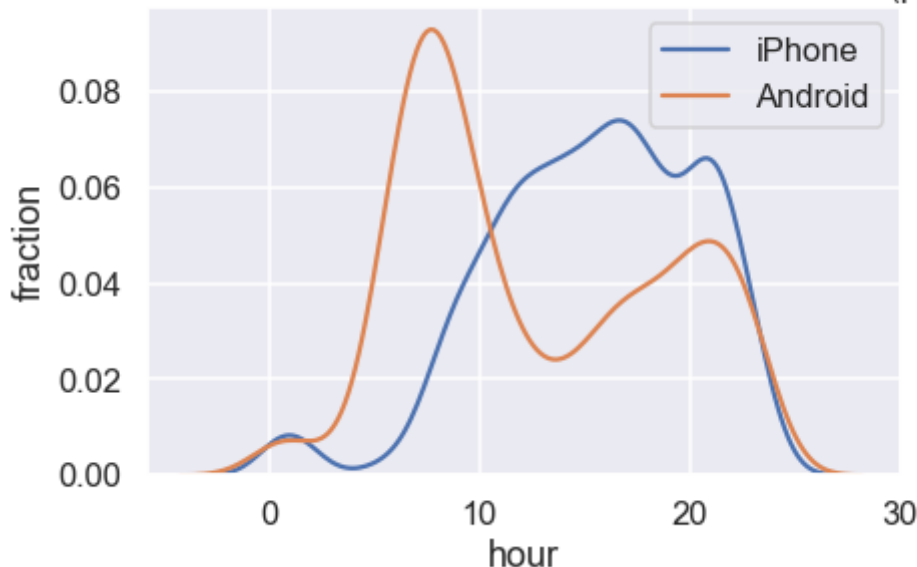
/Users/ryan\_kim/Documents/programming/ds/data/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

```
warnings.warn(msg, FutureWarning)
```

/Users/ryan\_kim/Documents/programming/ds/data/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. 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).

```
warnings.warn(msg, FutureWarning)
```

### 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:

I believe those two graph in 5b & 5c does not support Donald Trump's tweets from Android devices were written by him personally, and the tweets from iPhones were from his staff. **From the graphs 5b & 5c, the distribution of Android tweets is bimodal** and make us assume a pattern of tweets : in the morning, mid night, and afternoon. In addition, the structure of the distribution of Android tweets is not changed compared to 5b & 5c. I believe the fact suggests **the tweets from Android can be written with trump's staff**. The distribution of iphone is hard to check the pattern of usage, I guess Trump use iphone during the campaign (This is because it is more irregular for individuals to use their cell phones outside of work.)

## 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 [28]:

```
print(''.join(open("data/vader_lexicon.txt").readlines()[0:10]))
```

```
$: -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 [29]: # BEGIN YOUR CODE

sent = pd.read_csv("data/vader_lexicon.txt", delimiter="\t", header=None)
sent = sent[[0,1]].rename(columns={0:'token',1:'polarity'})
sent = sent.set_index('token')

END YOUR CODE
sent.head()
```

Out[29]:

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

```
In [30]: ok.grade("q6a");
```

```
~~~~~
Running tests
-----
Test summary
  Passed: 4
  Failed: 0
[ooooooooook] 100.0% passed
```

### 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 [31]:

```
# BEGIN SOLUTION
trump['text'] = trump['text'].str.lower()
# END SOLUTION
trump.head()
```

Out[31]:

|                           | time                   | source                    | text                                                                                                                                                                                                                                          | retweet_count |
|---------------------------|------------------------|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| <b>786204978629185536</b> | 2016-10-12<br>14:00:48 | Twitter<br>for<br>iPhone  | pay to play politics. \n#crookedhillary<br><a href="https://t.co/wjsl8itvvk">https://t.co/wjsl8itvvk</a>                                                                                                                                      | 24915         |
| <b>786201435486781440</b> | 2016-10-12<br>13:46:43 | Twitter<br>for<br>iPhone  | very little pick-up by the dishonest media<br>of incredible information provided by<br>wikileaks. so dishonest! rigged system!                                                                                                                | 22609         |
| <b>786189446274248704</b> | 2016-10-12<br>12:59:05 | Twitter<br>for<br>Android | crooked hillary clinton likes to talk about<br>the things she will do but she has been<br>there for 30 years - why didn't she do<br>them?                                                                                                     | 18329         |
| <b>786054986534969344</b> | 2016-10-12<br>04:04:47 | Twitter<br>for<br>iPhone  | thank you florida- a movement that has<br>never been seen before and will never be<br>seen again. lets get out &...<br><a href="https://t.co/t9xm9wfdzi">https://t.co/t9xm9wfdzi</a>                                                          | 18789         |
| <b>786007502639038464</b> | 2016-10-12<br>00:56:06 | Twitter<br>for<br>iPhone  | join me thursday in florida &...<br>ohio!\nw west palm beach, fl at<br>noon:\n <a href="https://t.co/jwbznqhxg9">https://t.co/jwbznqhxg9</a> \ncincinnati,<br>oh this 7:30pm:\n <a href="https://t.co/5w2uhalpix">https://t.co/5w2uhalpix</a> | 7761          |

In [32]:

```
ok.grade("q6b");
```

~~~~~  
Running tests

Test summary

Passed: 1

Failed: 0

[ooooooooook] 100.0% passed

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 [33]: # BEGIN YOUR CODE
# -----
# any character that isn't a Unicode word character or a whitespace character.
punct_re = r'^[\w\s]#[^a-zA-Z\s\d] is also possible
trump['no_punc'] = trump['text'].str.replace(punct_re, ' ', regex=True)
# -----
# END YOUR CODE
```

```
In [34]: ok.grade("q6c");
```

```
~~~~~
Running tests

-----
Test summary
  Passed: 10
  Failed: 0
[ooooooooook] 100.0% passed
```

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 [35]:

```
# BEGIN YOUR CODE
# -----
tidy_format = trump['no_punc'].str.split(expand=True)
tidy_format = tidy_format.stack()
tidy_format = tidy_format.reset_index(level=1)
tidy_format.columns = ['num', 'word']
# -----
# END YOUR CODE
tidy_format.head()
```

Out[35]:

	num	word
786204978629185536	0	pay
786204978629185536	1	to
786204978629185536	2	play
786204978629185536	3	politics
786204978629185536	4	crookedhillary

In [36]:

```
ok.grade("q6d");
```

~~~~~  
Running tests

-----  
Test summary

Passed: 2

Failed: 0

[ooooooooook] 100.0% passed

## 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 [37]:

```
# BEGIN YOUR CODE
# -----
# token의 종류가 7000여개, 그리고 tidy_format에서 트위터 문자 포맷 값이 25만개가 있음
# 요소들의 값을 합칠 때, 어떻게 서로 다른 갯수의 값을 합쳐줄 것인가?
trump['polarity'] = pd.merge(tidy_format,
                             sent,
                             left_on = 'word',
                             right_index = True)['polarity'].reset_index().group
trump['polarity'].replace(np.float64('nan'), 0, inplace = True)
# -----
# END YOUR CODE
trump[['text', 'polarity']]
```

Out[37]:

|                     |                                                                                                                                                                                                                                                                          | text | polarity |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|----------|
| 786204978629185536  | pay to play politics. \n#crookedhillary https://t.co/wjsl8itvvk                                                                                                                                                                                                          |      | 1.0      |
| 786201435486781440  | very little pick-up by the dishonest media of incredible information provided by wikileaks. so dishonest! rigged system!                                                                                                                                                 |      | -6.9     |
| 786189446274248704  | crooked hillary clinton likes to talk about the things she will do but she has been there for 30 years - why didn't she do them?                                                                                                                                         |      | 1.8      |
| 786054986534969344  | thank you florida- a movement that has never been seen before and will never be seen again. lets get out &... https://t.co/t9xm9wfdzi                                                                                                                                    |      | 1.5      |
| 786007502639038464  | join me thursday in florida & ohio!\nwest palm beach, fl at noon:\nhttps://t.co/jwbznqhxg9\ncincinnati, oh this 7:30pm:\nhttps://t.co/5w2uhalpix                                                                                                                         |      | 1.2      |
| ...                 | ...                                                                                                                                                                                                                                                                      | ...  | ...      |
| 1052344365442588673 | i will be interviewed tonight by trish regan on @foxbusiness at 8:00 p.m., right after the great lou dobbs!                                                                                                                                                              |      | 3.1      |
| 1052340597900627968 | rt @foxbusiness: tonight: @potus talks 2020 election in an exclusive interview with @trish_regan. don't miss the full interview on 'trish r...                                                                                                                           |      | -0.1     |
| 1052270676982333442 | wow, john james is making headway in michigan. we are bringing jobs back to the state, and the people of michigan appreciate it. debbie stabenow has been no help, if anything, a major hindrance. john james is a star, i hope the voters see it. polls are tightening! |      | 5.2      |
| 1052268013876064256 | ...during the call, and told me that he has already started, and will rapidly expand, a full and complete investigation into this matter. answers will be forthcoming shortly.                                                                                           |      | 1.4      |
| 1052268011900555265 | 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...                                                                                           |      | -1.9     |

10370 rows × 2 columns

In [38]:

```
ok.grade("q6e");
```

~~~~~  
Running tests

```
-----
Test summary
  Passed: 6
  Failed: 0
[ooooooooook] 100.0% passed
```

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 [39]:

```
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 aliens with criminal records, including those charged or convicted of 100,000 assaults, 30,000 sex crimes & 4000 violent killings." america's southern....

it is outrageous that poisonous synthetic heroin fentanyl comes pouring into the u.s. postal system from china. we can, and must, end this now! the senate should pass the stop act – and firmly stop this poison from killing our children and destroying our country. 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 for 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 anti-semitism from our world. we must unite to conquer hate.

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

In [40]:

```
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 team. a big win also for the great state of south carolina. look forward to seeing the team, and their brilliant coach, for the second 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 w

e 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. economy is roaring. supreme court pick getting great reviews. new poll says trump, at over 90%, is the most popular republican in history of the party. wow!

thank you, @wvgovernor jim justice, for that warm introduction. tonight, it was my great honor to attend the "greenbrier classic – salute to service dinner" in west virginia! god bless our veterans. god bless america – and happy independence day to all! <https://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.

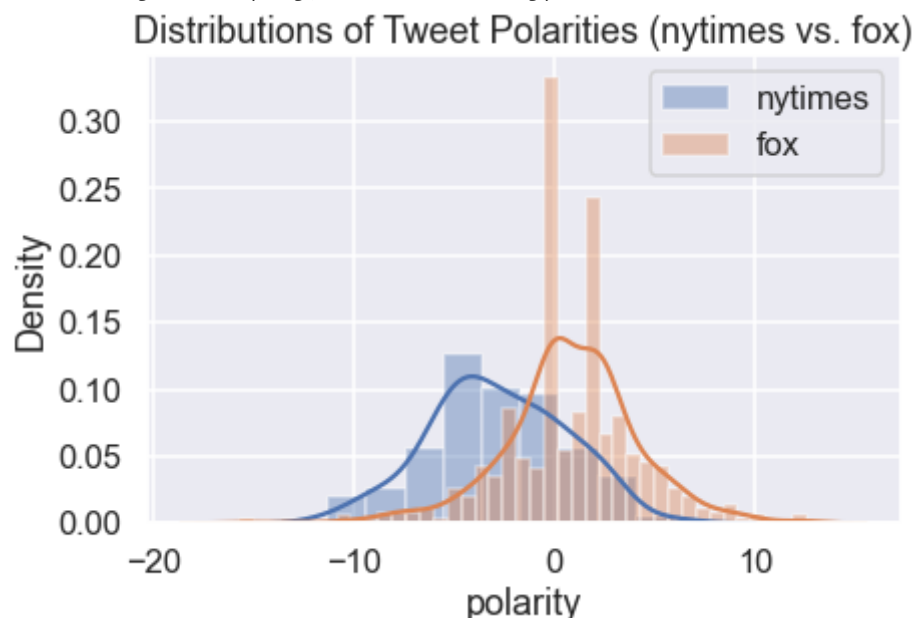
```
In [41]: sns.distplot(trump[trump['text'].str.lower().str.contains("nytimes")]['polarity'],
sns.distplot(trump[trump['text'].str.lower().str.contains("fox")]['polarity'], 1
plt.title('Distributions of Tweet Polarities (nytimes vs. fox)')
plt.legend();
```

/Users/ryan_kim/Documents/programming/ds/data/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

/Users/ryan_kim/Documents/programming/ds/data/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)



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!