COSE471 hw2

GAOZHONGSONG

TOTAL POINTS

23 / 23

QUESTION 1

1 Q1a **2 / 2**

✓ **+ 2 pts** **Pass all unit tests**

+ 0 pts Failed to pass all unit tests

QUESTION 2

2 Q1b **1 / 1**

✓ **+ 1 pts** **Pass all unit tests**

+ 0 pts Failed to pass all unit tests

QUESTION 3

3 Q1c **2 / 2**

✓ **+ 2 pts** **Pass all unit tests**

+ 0 pts Failed to pass unit tests

QUESTION 4

4 Q2 **2 / 2**

✓ **+ 2 pts** **Pass all unit tests**

+ 0 pts wrong

QUESTION 5

5 Q3 **1 / 1**

✓ **+ 1 pts** **Pass all unit tests**

+ 0 pts Failed to pass unit tests

QUESTION 6

6 Q4 **2 / 2**

- 0.5 pts No legend

- 0.5 pts No x and y axis labels

- 0.5 pts No title

- 0.5 pts Distributions are wrong

✓ **- 0 pts** **Correct**

QUESTION 7

7 Q5a **1 / 1**

✓ **+ 1 pts** **Pass all unit tests**

+ 0 pts Failed to pass unit tests.

QUESTION 8

8 Q5b **2 / 2**

- 0.5 pts No legend

- 0.5 pts No title

- 1 pts Distributions are wrong

✓ **- 0 pts** **Correct**

QUESTION 9

9 Q5c **2 / 2**

- 0.5 pts No legend

- 0.5 pts No x and y labels

- 1 pts Distributions are wrong

✓ **- 0 pts** **Correct**

QUESTION 10

10 Q6a **1 / 1**

✓ **+ 1 pts** **Pass all unit tests**

+ 0 pts wrong

QUESTION 11

11 Q6b **1 / 1**

✓ **+ 1 pts** **Pass all unit tests**

+ 0 pts wrong

QUESTION 12

12 Q6c **1 / 1**

✓ **+ 1 pts** **Pass all unit tests**

+ 0 pts Failed to pass all unit tests

QUESTION 13

13 Q6d **2 / 2**

✓ **+ 2 pts** **Pass all unit tests**

+ 0 pts wrong

QUESTION 14

14 Q6e **2 / 2**

✓ **+ 2 pts** **Pass all unit tests**

+ 0 pts wrong

QUESTION 15

15 Q5d **1 / 1**

✓ **+ 1 pts** **Correct**

+ 0 pts wrong

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 (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.replace.html) 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`.

In [236]:

```
# 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'}})
# ------------------------
# END YOUR CODE
bike.head()
```

Out[236]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit
0	1	2011-01-01	1	0	1	0	no	Sat	no	Clear
1	2	2011-01-01	1	0	1	1	no	Sat	no	Clear
2	3	2011-01-01	1	0	1	2	no	Sat	no	Clear
3	4	2011-01-01	1	0	1	3	no	Sat	no	Clear
4	5	2011-01-01	1	0	1	4	no	Sat	no	Clear

In [237]:

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

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

---------------------------------------------------------------------------------
Test summary
    Passed: 10
    Failed: 0
[oooooooooo] 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`

1 Q1a **2 / 2**

✓ **+ 2 pts** **Pass all unit tests**

+ 0 pts Failed to pass all unit tests

In [236]:

```
# 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'}})
# ------------------------
# END YOUR CODE
bike.head()
```

Out[236]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit
0	1	2011-01-01	1	0	1	0	no	Sat	no	Clear
1	2	2011-01-01	1	0	1	1	no	Sat	no	Clear
2	3	2011-01-01	1	0	1	2	no	Sat	no	Clear
3	4	2011-01-01	1	0	1	3	no	Sat	no	Clear
4	5	2011-01-01	1	0	1	4	no	Sat	no	Clear

In [237]:

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

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

---------------------------------------------------------------------------------
Test summary
    Passed: 10
    Failed: 0
[oooooooooo] 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 [238]:

```
num_holidays = len(bike[bike['holiday']=='yes'])
```

In [239]:

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

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

_____________________________________________________________________
Test summary
    Passed: 2
    Failed: 0
[oooooooooook] 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 (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.core.groupby.DataFrameGroupBy.agg.html) 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.

2 Q1b 1 / 1

✓ **+ 1 pts** **Pass all unit tests**

+ 0 pts Failed to pass all unit tests

In [238]:

```
num_holidays = len(bike[bike['holiday']=='yes'])
```

In [239]:

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

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

_____________________________________________________________________
Test summary
    Passed: 2
    Failed: 0
[oooooooooook] 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 (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.core.groupby.DataFrameGroupBy.agg.html) 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.

3 Q1c 2 / 2

✓ **+ 2 pts** **Pass all unit tests**

+ 0 pts Failed to pass unit tests

```
50087458',
          'profile_image_url': 'http://pbs.twimg.com/profile_images/87427619735759
6672/kUuht00m_normal.jpg',
          'profile_image_url_https': 'https://pbs.twimg.com/profile_images/8742761
97357596672/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/0MxB0x7xC5',
          '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 as below.**

	time	source	text	retweet_count
690171032150237184	2016-01-21 13:56:11	Twitter for Android	"@bigop1: @realDonaldTrump @SarahPalinUSA https://t.co/3kYQGqeVyD"	1059
690171403388104704	2016-01-21 13:57:39	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	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	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	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

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

In [244]:

```
# BEGIN YOUR CODE
# ------------------------
id = [i['id'] for i in all_tweets]
time = [pd.to_datetime(i['created_at']) for i in all_tweets]
source = [i['source'] for i in all_tweets]
text = [i['text']
        if 'text' in i
        else i['full_text']
        for i in all_tweets]
retweet_count = [i['retweet_count'] for i in all_tweets]
trump = pd.DataFrame({'time' : time, 'source' : source, 'text' : text, 'retweet_count' : retweet
_count},
                     index = id)
# ------------------------
# END YOUR CODE
trump.head()
```

Out[244]:

	time	source	
786204978629185536	2016-10-12 14:00:48	Twitter for iPhone	PAY TO PLAY I https://t.co/wjsl
786201435486781440	2016-10-12 13:46:43	Twitter for iPhone	Very little pick-u incredible infor WikiLeaks. So
786189446274248704	2016-10-12 12:59:05	Twitter for Android	Crooked Hillary the things she there for 30 ye them?
786054986534969344	2016-10-12 04:04:47	Twitter for iPhone	Thank you Flo never been se seen again. Le https://t.co/t9XI
786007502639038464	2016-10-12 00:56:06	Twitter for iPhone	Join me Thursd Ohio!\nWest P noon:\nhttps://t OH this 7:30pm

In [245]:

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

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

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

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 [246]:

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

Out[246]:

```
array([ '<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for i
Phone</a>',
       '<a href="http://twitter.com/download/android" rel="nofollow">Twitter for A
ndroid</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 i
Pad</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/).

4 Q2 2 / 2

✓ **+ 2 pts** **Pass all unit tests**

+ 0 pts wrong

In [245]:

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

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

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

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 [246]:

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

Out[246]:

```
array([ '<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for i
Phone</a>',
       '<a href="http://twitter.com/download/android" rel="nofollow">Twitter for A
ndroid</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 i
Pad</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 [247]:

```
# BEGIN YOUR CODE
# ------------------------
trump['source'] = trump['source'].str.replace(r"₩<.*?₩>", "")
# ------------------------
# END YOUR CODE
```

In [248]:

```
ok.grade("q3");
```

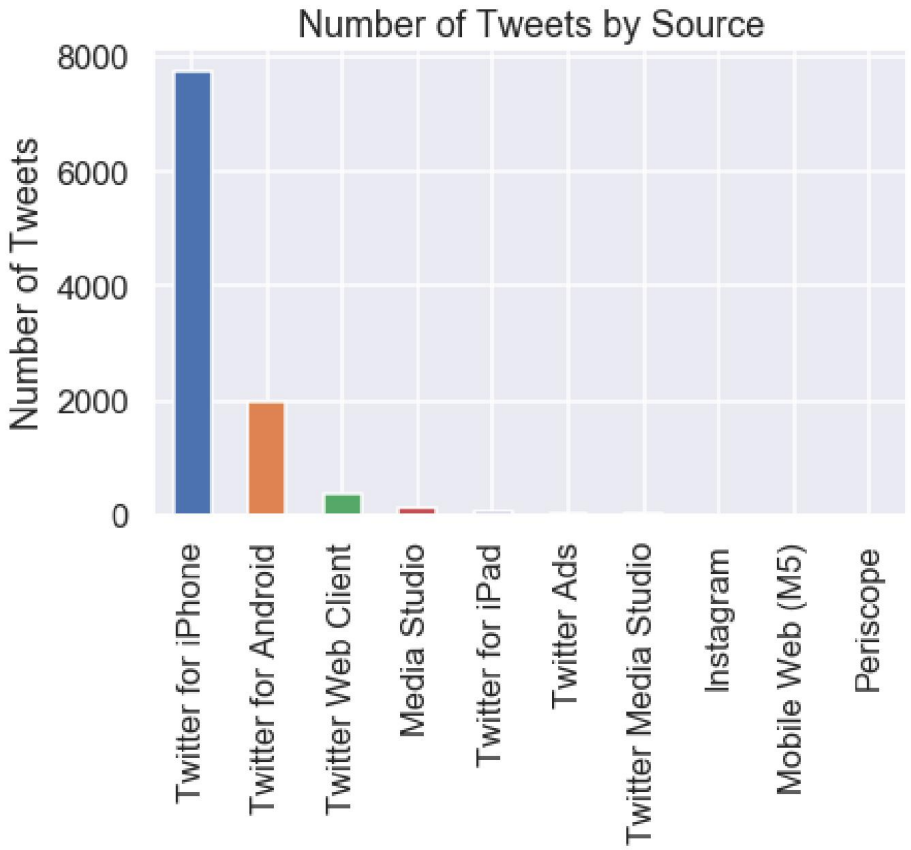
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

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

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

In [249]:

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



5 Q3 1 / 1

✓ **+ 1 pts** **Pass all unit tests**

+ 0 pts Failed to pass unit tests

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.

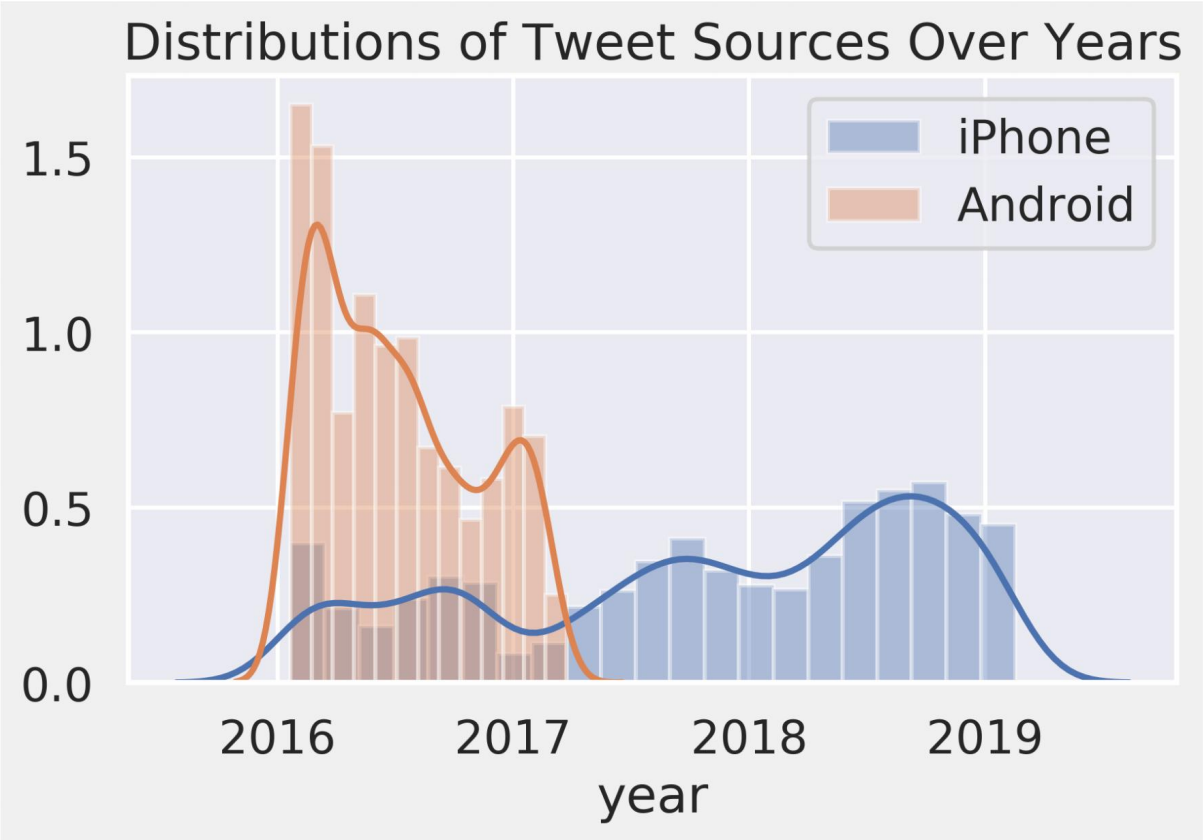
(Code borrowed from https://stackoverflow.com/questions/6451655/python-how-to-convert-datetime-dates-to-decimal-years (https://stackoverflow.com/questions/6451655/python-how-to-convert-datetime-dates-to-decimal-years))

In [250]:

```
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:



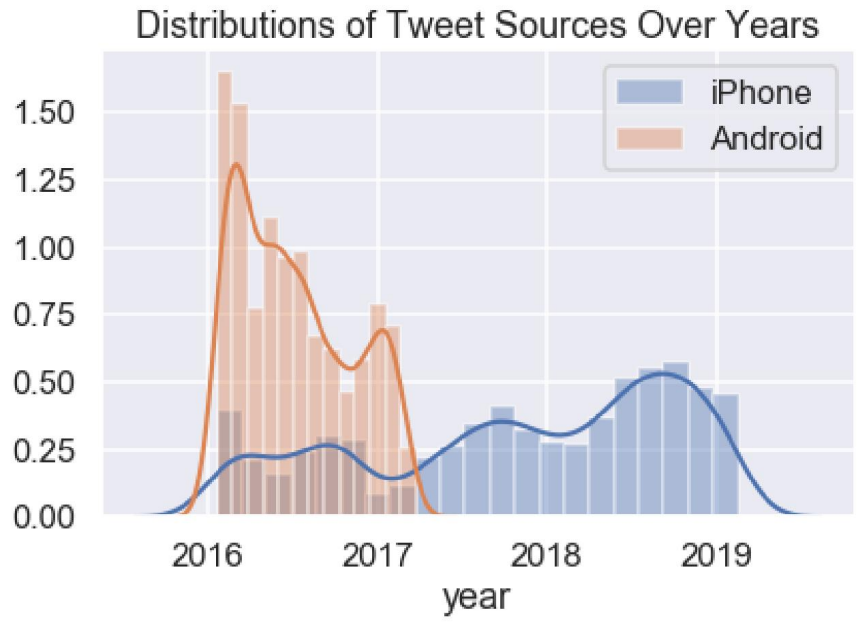
In [251]:

```
# BEGIN YOUR CODE
# ------------------------
trump['source'].value_counts().head(2).index

for thissource in trump['source'].value_counts().head(2).index:
    sns.distplot(trump[trump['source'] == thissource]['year'], label = thissource[12: ], hist =
True)
plt.legend()
plt.title('Distributions of Tweet Sources Over Years')
# ------------------------
# END YOUR CODE
```

Out[251]:

Text(0.5,1,'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 (https://www.wikiwand.com/en/List_of_UTC_time_offsets) (notice the +0000 in the first few tweets).

6 Q4 2 / 2

- **0.5 pts** No legend
- **0.5 pts** No x and y axis labels
- **0.5 pts** No title
- **0.5 pts** Distributions are wrong

✓ **- 0 pts** **Correct**

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 (https://pandas.pydata.org/pandas-docs/stable/getting_started/basics.html#basics-dt-accessors).

In [254]:

```
# BEGIN YOUR CODE
# -----------------------
trump['hour'] = trump['est_time'].dt.hour + (trump['est_time'].dt.minute / 60) + (trump['est_tim
e'].dt.second / (60 * 60))
# -----------------------
# END YOUR CODE
```

In [255]:

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

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

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

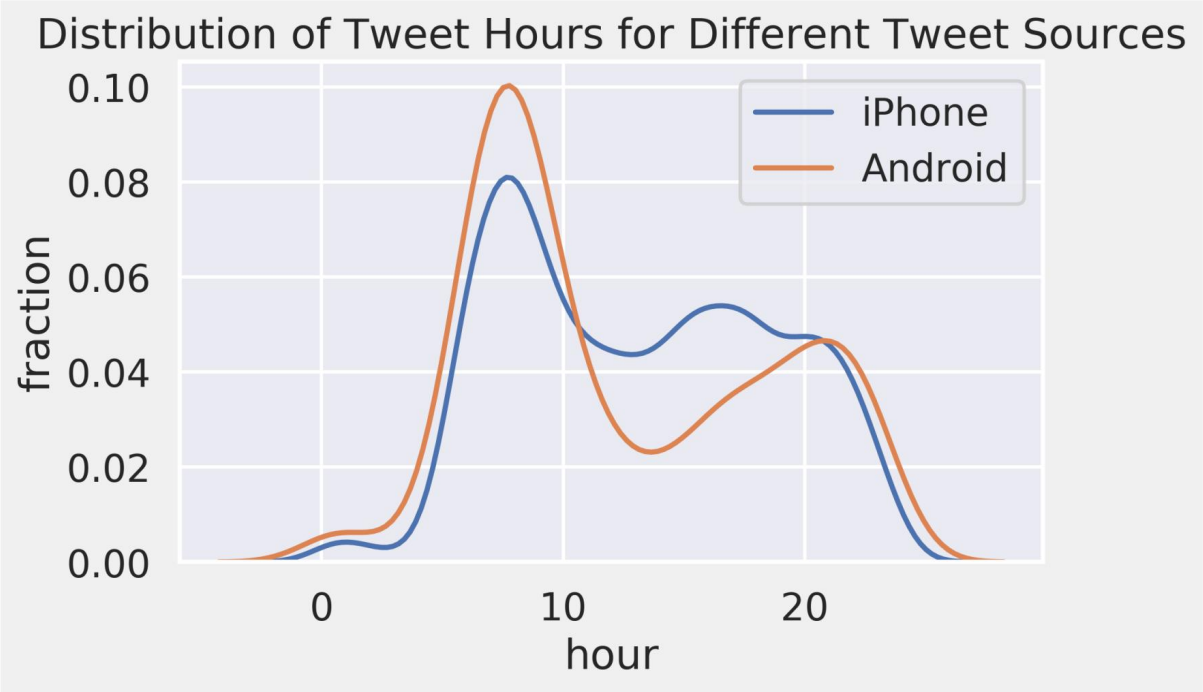
7 Q5a 1 / 1

✓ **+ 1 pts** **Pass all unit tests**

+ 0 pts Failed to pass 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. Your plot should look similar to the following:

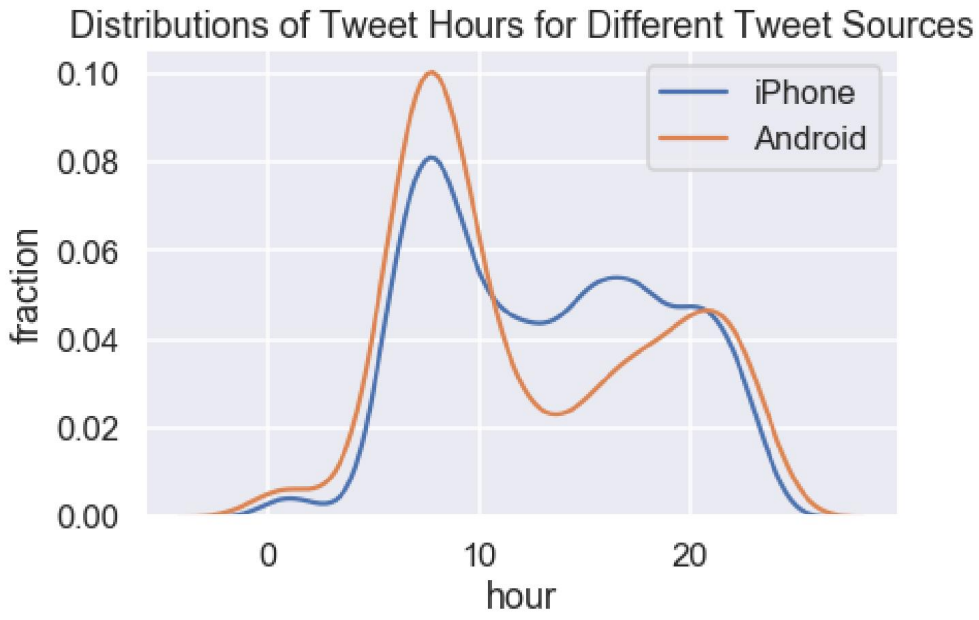


In [256]:

```
# BEGIN YOUR CODE
# ------------------------
for thissource in trump['source'].value_counts().head(2).index:
    sns.distplot(trump[trump['source'] == thissource]['hour'], label = thissource[12: ], hist =
False)
plt.legend()
plt.title('Distributions of Tweet Hours for Different Tweet Sources')
plt.ylabel('fraction')
# ------------------------
# END YOUR CODE
```

Out[256]:

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



8 Q5b **2 / 2**

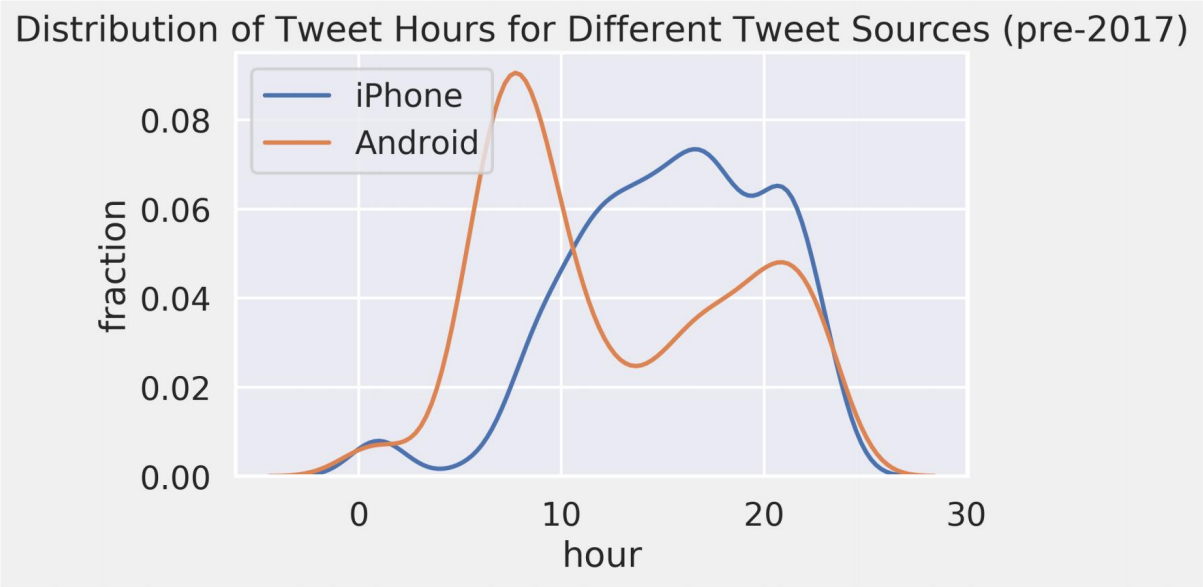
- **0.5 pts** No legend
- **0.5 pts** No title
- **1 pts** Distributions are wrong

✓ **- 0 pts** **Correct**

Question 5c

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

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:



9 Q5c 2 / 2

- **0.5 pts** No legend
- **0.5 pts** No x and y labels
- **1 pts** Distributions are wrong

✓ **- 0 pts Correct**

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

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

In [258]:

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

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

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 [281]:

```
# BEGIN YOUR CODE
# ------------------------
sent = pd.read_csv('data/vader_lexicon.txt', sep='₩t',
                   usecols=[0, 1], header=None, names=['token', 'polarity'],
                   index_col='token')
# ------------------------
# END YOUR CODE
sent.head()
```

Out[281]:

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

In [282]:

```
ok.grade("q6a");
```

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

---------------------------------------------------------------------------
Test summary
    Passed: 4
    Failed: 0
[oooooooooo] 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.

10 Q6a **1 / 1**

✓ **+ 1 pts** **Pass all unit tests**

+ 0 pts wrong

In [281]:

```
# BEGIN YOUR CODE
# ------------------------
sent = pd.read_csv('data/vader_lexicon.txt', sep='₩t',
                   usecols=[0, 1], header=None, names=['token', 'polarity'],
                   index_col='token')
# ------------------------
# END YOUR CODE
sent.head()
```

Out[281]:

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

In [282]:

```
ok.grade("q6a");
```

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

---------------------------------------------------------------------------
Test summary
    Passed: 4
    Failed: 0
[oooooooooo] 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 [283]:

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

Out[283]:

	time	source	text	retw
786204978629185536	2016-10-12 14:00:48	Twitter for iPhone	pay to play politics. \n#crookedhillary https://t.co/wjsl8itvvk	2491
786201435486781440	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!	2260
786189446274248704	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?	1832
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 &… https://t.co/t9xm9wfdzi	1878
786007502639038464	2016-10-12 00:56:06	Twitter for iPhone	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	7761

In [284]:

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

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

---------------------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 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 [285]:

```
# BEGIN YOUR CODE
# ------------------------
punct_re = r'[^\w\s]'  # Save your regex in punct_re
trump['no_punc'] = trump['text'].str.replace(punct_re, ' ')
# ------------------------
# END YOUR CODE
```

In [286]:

```
ok.grade("q6c");
```

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

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

11 Q6b **1 / 1**

✓ **+ 1 pts** **Pass all unit tests**

+ 0 pts wrong

In [284]:

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

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

---------------------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 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 [285]:

```
# BEGIN YOUR CODE
# ------------------------
punct_re = r'[^\w\s]'  # Save your regex in punct_re
trump['no_punc'] = trump['text'].str.replace(punct_re, ' ')
# ------------------------
# END YOUR CODE
```

In [286]:

```
ok.grade("q6c");
```

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

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

12 Q6c **1 / 1**

✓ **+ 1 pts** **Pass all unit tests**

+ 0 pts Failed to pass all unit tests

Question 6d

Now, let's convert the tweets into what's called a *tidy format* (https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html) to make the sentiments easier to calculate. Use the `no_punc` column of `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.

13 Q6d **2 / 2**

✓ **+ 2 pts** **Pass all unit tests**

+ 0 pts wrong

In [287]:

```
# BEGIN YOUR CODE
# ------------------------
tidy_format = trump['no_punc'].str.split(expand = True).stack().reset_index(level = 1).rename(c
olumns = {'level_1': 'num', 0: 'word'})
# ------------------------
# END YOUR CODE
tidy_format.head()
```

Out[287]:

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

In [288]:

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

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

---------------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[oooooooooook] 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 [289]:

```
# BEGIN YOUR CODE
# ----------------------
trump['polarity'] = (
    tidy_format
    .merge(sent, how='left', left_on='word', right_index=True)
    .reset_index()
    .loc[:, ['index', 'polarity']]
    .groupby('index')
    .sum()
    .fillna(0)
)
# ----------------------
# END YOUR CODE
trump[['text', 'polarity']].head()
```

Out[289]:

	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

In [290]:

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

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

---------------------------------------------------------------------

Test summary
    Passed: 6
    Failed: 0
[oooooooooo] 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:

14 Q6e **2 / 2**

✓ **+ 2 pts** **Pass all unit tests**

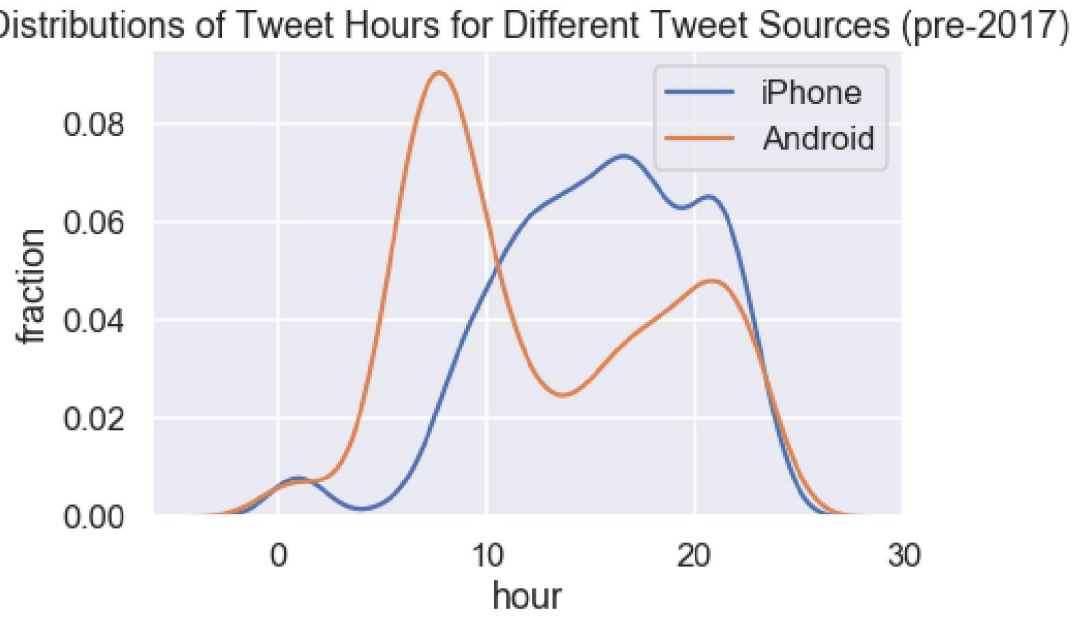
+ 0 pts wrong

In [257]:

```
# BEGIN YOUR CODE
# ------------------------
pre_2017 = trump[trump['est_time'].dt.year < 2017]
for thissource in pre_2017['source'].value_counts().head(2).index:
    sns.distplot(pre_2017[pre_2017['source'] == thissource]['hour'], label = thissource[12: ], h
ist = False)
plt.legend()
plt.title('Distributions of Tweet Hours for Different Tweet Sources (pre-2017)')
plt.ylabel('fraction')
# ------------------------
# END YOUR CODE
```

Out[257]:

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



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: yes, figure give support to this theory, Android phone has a higher value around maybe 8:30 and 21:00, which is about the time before and after work. Assitional analysis that could help can be the specific time he go to work and finish work.

15 Q5d **1 / 1**

✓ **+ 1 pts** **Correct**

+ 0 pts wrong