Wrangle and Analyze Data Project Details

1. `wrangle_act.ipynb` : code for gathering, assessing, cleaning, analyzing, and visualizing data
2. `wrangle_report.pdf` or `wrangle_report.html` : `300-600` word documentation for data wrangling steps: gather, assess, and clean
3. `act_report.pdf` or `act_report.html` : `250` word minimum documentation of analysis and insights into final data
4. `twitter_archive_enhanced.csv` : file as given
5. `image_predictions.tsv` : file downloaded programmatically
6. `tweet_json.txt` : file constructed via API
7. `twitter_archive_master.csv` : combined and cleaned data
8. any additional files
9. At least three `3` insights and one `1` visualization must be assessed.

Gathering

In [5]:
```
import pandas as pd
import requests
import os
import logging
import sys
import json
# global logger level is configured in main()
Logger = None

df = pd.read_csv('twitter-archive-enhanced.csv')
```

Requests Library

In [17]:
```
folder_name = 'tweet_image_predictions'
if not os.path.exists(folder_name):
    os.makedirs(folder_name)

url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv'
response = requests.get(url)

with open(os.path.join(folder_name, url.split('/')[-1]), mode='wb') as file:
    file.write(response.content)
```

Twitter API

1. Query all of the tweet IDs in the WeRateDogs Twitter archive, printing out each tweet ID after it was queried.
2. Set the wait_on_rate_limit and wait_on_rate_limit_notify parameters to True in the tweepy.api class.
3. Tweet data is stored in JSON format by Twitter.
4. Set the tweet_mode parameter to 'extended' in the get_status call, i.e., api.get_status(tweet_id, tweet_mode='extended').

- You only want original ratings (no retweets) that have images.
- Though there are 5000+ tweets in the dataset, not all are dog ratings and some are retweets.

In [21]:
```
import tweepy
from tweepy import OAuthHandler
import json
from timeit import default_timer as timer

consumer_key = 'API_KEY'
consumer_secret = 'SECRET'
access_token = 'TOKEN'
access_secret = 'SECRET'

auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api = tweepy.API(auth_handler=auth, wait_on_rate_limit=True, wait_on_rate_limit_n
otify=True)
```

In [25]:

```
tweet_ids = df.tweet_id.values
len(tweet_ids)

count = 0
fails_dict = {}
start = timer()
with open('tweet_json.txt', 'w') as outfile:
    for tweet_id in tweet_ids:
        count += 1
        print(str(count) + ": " + str(tweet_id))
        try:
            tweet = api.get_status(tweet_id, tweet_mode='extended')
            print("Got thru!")
            json.dump(tweet._json, outfile)
            outfile.write('\n')
        except tweepy.TweepError as e:
            print("Did not get thru.")
            fails_dict[tweet_id] = e
            pass
end = timer()
print(end - start)
print(fails_dict)
```

```
2327: 666411507551481857
Got thru!
2328: 666407126856765440
Got thru!
2329: 666396247373291520
Got thru!
2330: 666373753744588802
Got thru!
2331: 666362758909284353
Got thru!
2332: 666353288456101888
Got thru!
2333: 666345417576210432
Got thru!
2334: 666337882303524864
Got thru!
2335: 666293911632134144
Got thru!
2336: 666287406224695296
Got thru!
2337: 666273097616637952
Got thru!
2338: 666268910803644416
Got thru!
2339: 666104133288665088
Got thru!
2340: 666102155909144576
Got thru!
2341: 666099513787052032
Got thru!
2342: 666094000022159362
Got thru!
2343: 666082916733198337
Got thru!
2344: 666073100786774016
Got thru!
2345: 666071193221509120
Got thru!
2346: 666063827256086533
Got thru!
2347: 666058600524156928
Got thru!
2348: 666057090499244032
Got thru!
2349: 666055525042405380
Got thru!
2350: 666051853826850816
Got thru!
2351: 666050758794694657
Got thru!
2352: 666049248165822465
Got thru!
2353: 666044226329800704
Got thru!
2354: 666033412701032449
Got thru!
2355: 666029285002620928
Got thru!
2356: 666020888022790149
Got thru!
1915.0742060539997
{888202515573088257: TweepError([{'code': 144, 'message': 'No status found with
```

```
that ID.'}]), 873697596434513921: TweepError([{'code': 144, 'message': 'No statu
s found with that ID.'}]), 872668790621863937: TweepError([{'code': 144, 'messag
e': 'No status found with that ID.'}]), 872261713294495745: TweepError([{'code':
144, 'message': 'No status found with that ID.'}]), 869988702071779329: TweepErr
or([{'code': 144, 'message': 'No status found with that ID.'}]), 866816280283807
744: TweepError([{'code': 144, 'message': 'No status found with that ID.'}]), 86
1769973181624320: TweepError([{'code': 144, 'message': 'No status found with tha
t ID.'}]), 856602993587888130: TweepError([{'code': 144, 'message': 'No status f
ound with that ID.'}]), 851953902622658560: TweepError([{'code': 144, 'message':
'No status found with that ID.'}]), 845459076796616705: TweepError([{'code': 14
4, 'message': 'No status found with that ID.'}]), 844704788403113984: TweepError
([{'code': 144, 'message': 'No status found with that ID.'}]), 842892208864923648: TweepError([{'code': 144, 'message': 'No status found with that ID.'}]), 8373
66284874571778: TweepError([{'code': 144, 'message': 'No status found with that
ID.'}]), 837012587749474308: TweepError([{'code': 144, 'message': 'No status fou
nd with that ID.'}]), 829374341691346946: TweepError([{'code': 144, 'message':
'No status found with that ID.'}]), 827228250799742977: TweepError([{'code': 14
4, 'message': 'No status found with that ID.'}]), 812747805718642688: TweepError
([{'code': 144, 'message': 'No status found with that ID.'}]), 802247111496568832: TweepError([{'code': 144, 'message': 'No status found with that ID.'}]), 7791
23168116150273: TweepError([{'code': 144, 'message': 'No status found with that
ID.'}]), 775096608509886464: TweepError([{'code': 144, 'message': 'No status fou
nd with that ID.'}]), 770743923962707968: TweepError([{'code': 144, 'message':
'No status found with that ID.'}]), 754011816964026368: TweepError([{'code': 14
4, 'message': 'No status found with that ID.'}]), 680055455951884288: TweepError
([{'code': 144, 'message': 'No status found with that ID.'}])}
```

In [26]:
```
df.head()
```

Out[26]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	s
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000	href="http://twitter.com/download/ip
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitter.com/download/ip
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitter.com/download/ip
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	href="http://twitter.com/download/ip
4	891327558926688256	NaN	NaN	2017-07-29 16:00:24 +0000	href="http://twitter.com/download/ip

In [33]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

In [34]:
```
df.describe()
```

Out[34]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	retweeted_status_id	retweeted_status_user_id
count	2.356000e+03	7.800000e+01	7.800000e+01	1.810000e+02	1.810000e+02
mean	7.427716e+17	7.455079e+17	2.014171e+16	7.720400e+17	1.241698e+16
std	6.856705e+16	7.582492e+16	1.252797e+17	6.236928e+16	9.599254e+16
min	6.660209e+17	6.658147e+17	1.185634e+07	6.661041e+17	7.832140e+05
25%	6.783989e+17	6.757419e+17	3.086374e+08	7.186315e+17	4.196984e+09
50%	7.196279e+17	7.038708e+17	4.196984e+09	7.804657e+17	4.196984e+09
75%	7.993373e+17	8.257804e+17	4.196984e+09	8.203146e+17	4.196984e+09
max	8.924206e+17	8.862664e+17	8.405479e+17	8.874740e+17	7.874618e+17

In [6]:
```
with open('tweet_json.txt') as file:
    tweet_json_list = []
    for line in file:
        tweet_json_list.append(json.loads(line)) # cite 1
```

In [7]:
```
tweet_json_list[0]
```

Out[7]:

```
{'created_at': 'Tue Aug 01 16:23:56 +0000 2017',
 'id': 892420643555336193,
 'id_str': '892420643555336193',
 'full_text': "This is Phineas. He's a mystical boy. Only ever appears in the ho
le of a donut. 13/10 https://t.co/MgUWQ76dJU",
 'truncated': False,
 'display_text_range': [0, 85],
 'entities': {'hashtags': [],
  'symbols': [],
  'user_mentions': [],
  'urls': [],
  'media': [{'id': 892420639486877696,
    'id_str': '892420639486877696',
    'indices': [86, 109],
    'media_url': 'http://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg',
    'media_url_https': 'https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg',
    'url': 'https://t.co/MgUWQ76dJU',
    'display_url': 'pic.twitter.com/MgUWQ76dJU',
    'expanded_url': 'https://twitter.com/dog_rates/status/892420643555336193/pho
to/1',
    'type': 'photo',
    'sizes': {'thumb': {'w': 150, 'h': 150, 'resize': 'crop'},
     'medium': {'w': 540, 'h': 528, 'resize': 'fit'},
     'small': {'w': 540, 'h': 528, 'resize': 'fit'},
     'large': {'w': 540, 'h': 528, 'resize': 'fit'}}}]},
 'extended_entities': {'media': [{'id': 892420639486877696,
    'id_str': '892420639486877696',
    'indices': [86, 109],
    'media_url': 'http://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg',
    'media_url_https': 'https://pbs.twimg.com/media/DGKD1-bXoAAIAUK.jpg',
    'url': 'https://t.co/MgUWQ76dJU',
    'display_url': 'pic.twitter.com/MgUWQ76dJU',
    'expanded_url': 'https://twitter.com/dog_rates/status/892420643555336193/pho
to/1',
    'type': 'photo',
    'sizes': {'thumb': {'w': 150, 'h': 150, 'resize': 'crop'},
     'medium': {'w': 540, 'h': 528, 'resize': 'fit'},
     'small': {'w': 540, 'h': 528, 'resize': 'fit'},
     'large': {'w': 540, 'h': 528, 'resize': 'fit'}}}]},
 'source': '<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter
for iPhone</a>',
 '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,
 'in_reply_to_screen_name': None,
 'user': {'id': 4196983835,
  'id_str': '4196983835',
  'name': 'WeRateDogs™',
  'screen_name': 'dog_rates',
  'location': '「 DM YOUR DOGS 」',
  'description': 'Your Only Source For Professional Dog Ratings Instagram and Fa
cebook ➪ WeRateDogs partnerships@weratedogs.com ',
  'url': 'https://t.co/N7sNNHSfPq',
  'entities': {'url': {'urls': [{'url': 'https://t.co/N7sNNHSfPq',
     'expanded_url': 'http://weratedogs.com',
     'display_url': 'weratedogs.com',
     'indices': [0, 23]}]},
   'description': {'urls': []}},
  'protected': False,
  'followers_count': 8270851,
```

```
  'friends_count': 12,
  'listed_count': 6385,
  'created_at': 'Sun Nov 15 21:41:29 +0000 2015',
  'favourites_count': 142537,
  'utc_offset': None,
  'time_zone': None,
  'geo_enabled': True,
  'verified': True,
  'statuses_count': 10603,
  'lang': None,
  'contributors_enabled': False,
  'is_translator': False,
  'is_translation_enabled': False,
  'profile_background_color': '000000',
  'profile_background_image_url': 'http://abs.twimg.com/images/themes/theme1/bg.
png',
  'profile_background_image_url_https': 'https://abs.twimg.com/images/themes/the
me1/bg.png',
  'profile_background_tile': False,
  'profile_image_url': 'http://pbs.twimg.com/profile_images/1112594177961844736/
qQK8NJT-_normal.jpg',
  'profile_image_url_https': 'https://pbs.twimg.com/profile_images/1112594177961
844736/qQK8NJT-_normal.jpg',
  'profile_banner_url': 'https://pbs.twimg.com/profile_banners/4196983835/156460
0075',
  'profile_link_color': 'F5ABB5',
  'profile_sidebar_border_color': '000000',
  'profile_sidebar_fill_color': '000000',
  'profile_text_color': '000000',
  'profile_use_background_image': False,
  'has_extended_profile': False,
  'default_profile': False,
  'default_profile_image': False,
  'following': False,
  'follow_request_sent': False,
  'notifications': False,
  'translator_type': 'none'},
 'geo': None,
 'coordinates': None,
 'place': None,
 'contributors': None,
 'is_quote_status': False,
 'retweet_count': 7983,
 'favorite_count': 37274,
 'favorited': False,
 'retweeted': False,
 'possibly_sensitive': False,
 'possibly_sensitive_appealable': False,
 'lang': 'en'}
```


In [8]:
```
df_json = pd.DataFrame.from_records(tweet_json_list)
```

In [10]:
```
df_json.sample(10)
```

Out[10]:

	contributors	coordinates	created_at	display_text_range	entities	extended_entities	favorite
184	None	None	Sat Apr 22 16:18:34 +0000 2017	[0, 110]	{'hashtags': [], 'symbols': [], 'user_mentions...	NaN	
146	None	None	Thu May 11 17:34:13 +0000 2017	[0, 135]	{'hashtags': [], 'symbols': [], 'user_mentions...	{'media': [{'id': 862722516858445824, 'id_str'...	
490	None	None	Sat Dec 24 17:18:34 +0000 2016	[0, 94]	{'hashtags': [], 'symbols': [], 'user_mentions...	{'media': [{'id': 812709052820099072, 'id_str'...	
1243	None	None	Wed Mar 16 00:37:03 +0000 2016	[0, 140]	{'hashtags': [], 'symbols': [], 'user_mentions...	{'media': [{'id': 709901249051754496, 'id_str'...	
2125	None	None	Thu Nov 26 05:28:02 +0000 2015	[0, 129]	{'hashtags': [], 'symbols': [], 'user_mentions...	{'media': [{'id': 669749424290164736, 'id_str'...	
145	None	None	Fri May 12 00:46:44 +0000 2017	[0, 121]	{'hashtags': [], 'symbols': [], 'user_mentions...	{'media': [{'id': 862831346963447808, 'id_str'...	
966	None	None	Fri Jul 01 20:31:43 +0000 2016	[0, 112]	{'hashtags': [], 'symbols': [], 'user_mentions...	{'media': [{'id': 748977397119258624, 'id_str'...	
655	None	None	Sat Oct 22 00:45:17 +0000 2016	[0, 41]	{'hashtags': [], 'symbols': [], 'user_mentions...	{'media': [{'id': 789628648391401472, 'id_str'...	
31	None	None	Sat Jul 15 02:45:48 +0000 2017	[0, 50]	{'hashtags': [{'text': 'BATP', 'indices': [21,...	NaN	
1248	None	None	Mon Mar 14 18:42:20 +0000 2016	[0, 140]	{'hashtags': [], 'symbols': [], 'user_mentions...	{'media': [{'id': 709449595638706176, 'id_str'...	

10 rows × 32 columns

Assessing Data

- Assess and clean at least **eight `8` quality issues** and **two `2` tidiness issues** in this dataset.
- Cleaning includes merging individual pieces of data according to the rules of tidy data.
- The fact that the rating numerators are greater than the denominators does not need to be cleaned.
- This unique rating system is a big part of the popularity of WeRateDogs.
- You do not need to gather the tweets beyond August 1st, 2017.

Quality

twitter-archive-enhanced:

1. `NaN` values in `in_reply_to_status_id`, `in_reply_to_user_id`, `retweeted_status_id`, `retweeted_status_timestamp`
2. `None` values in `doggo`, `floofer`, `pupper`, `puppo`
3. Single letters in `name`

tweet_json.txt:

1. `None` values in `contributors`, `coordinates`, `geo`
2. `NaN` values in `extended_entities`, `quoted_status`, `quoted_status_id`, `quoted_status_id_str`, `quoted_status_permalink`, `retweeted_status`
3.
4.
5.

Tidiness

1. the `twitter-archive-enhanced` df and `tweet_json` df need to be merged to include the most important metadata that is readily available.
2. extract `user` from string object convert to string

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Storing, Analyzing, and Visualizing Data

In [27]:
```
from sqlalchemy import create_engine
engine = create_engine('sqlite:///tweets.db')
```

In [28]:
```
df.to_sql('master', engine, index=False)
```

In [29]:
```
df_gather = pd.read_sql('SELECT * FROM master', engine)
```

In [30]:
```
df_gather.head(3)
```

Out[30]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	s
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000	href="http://twitter.com/download/ip
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	href="http://twitter.com/download/ip
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	href="http://twitter.com/download/ip

In [ ]:

Citations:

1. #https://stackoverflow.com/questions/47889565/reading-json-objects-from-text-file-into-pandas (https://stackoverflow.com/questions/47889565/reading-json-objects-from-text-file-into-pandas)

In [ ]: