WeRateDogs Twitter Data from 2015 to 2017

Data Wrangling Project

Table of Contents

1. Data Gathering

1.1 Directly download the WeRateDogs Twitter archive data (twitter_archive_enhanced.csv)

In [1]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df_archive = pd.read_csv("twitter-archive-enhanced.csv")
```

1.2 Use the Requests library to download the tweet image prediction (image_predictions.tsv)

In [3]:

```
import requests

# Access file

url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_im
response = requests.get(url)
response
```

Out[3]:

```
<Response [200]>
```

In [4]:

```
# Test
response.content[:100]
```

Out[4]:

```
b'tweet_id\tjpg_url\timg_num\tp1\tp1_conf\tp1_dog\tp2\tp2_conf\tp2_dog\tp3\t
p3_conf\tp3_dog\n666020888022790149\tht'
```

In [5]:

```
import os
```

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

os.listdir(folder_name)
```

Out[5]:

```
['.DS_Store', 'image-predictions.tsv']
```

In [6]:

```
df_image = pd.read_csv("image-predictions.tsv",  sep='\t')
```

1.3 Use the Tweepy library to query additional data via the Twitter API (tweet_json.txt)

Note: Personal keys hidden

In [7]:

```
import json
import tweepy
import time

consumer_key = ''
consumer_secret = ''
access_token = ''
access_secret = ''
```

In [8]:

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)
api = tweepy.API(auth, wait_on_rate_limit=True)
```

In [9]:

```
# Tweet IDs for which to gather additional data via Twitter's API
tweet_ids = df_archive.tweet_id.values
len(tweet_ids)
```

Out[9]:

```
2356
```

In [ ]:

```
# Output is cleared for easier viewing in html/pdf

import json

df_tweets = pd.DataFrame(columns=["tweet ID", "retweet count", "favorite co
with open('tweet-json.txt') as file:
    for line in file:
        print(line)
        status=json.loads(line)
        tweet_id=status['id_str']
        rt_count=status['retweet_count']
        fav_count=status['favorite_count']
        df_tweets=df_tweets.append(pd.DataFrame([[tweet_id,rt_count,fav_cou
                                             columns=["tweet ID", "retwe
```

2. Assessing Data

2.1 Exploration

In [12]:

```
df_archive.sample(5)
```

Out[12]:

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	
176	857746408056729600	NaN	NaN	2017-04-28 00:00:54 +0000	href="htt
1459	695064344191721472	NaN	NaN	2016-02-04 02:00:27 +0000	href="htt
971	750101899009982464	NaN	NaN	2016-07-04 23:00:03 +0000	href="htt
1004	747816857231626240	NaN	NaN	2016-06-28 15:40:07 +0000	href="htt
2187	668979806671884288	NaN	NaN	2015-11-24 02:29:49 +0000	href="htt

In [13]:

```
df_image.sample(5)
```

Out[13]:

	tweet_id	jpg_url	img_num	
456	674774481756377088	https://pbs.twimg.com/media/CV1HztsWoAAuZwo.jpg	1	
1927	857989990357356544	https://pbs.twimg.com/media/C-gxV9ZXkAIBL-S.jpg	1	Fr
131	668297328638447616	https://pbs.twimg.com/media/CUZE4IWW4AAZmDf.jpg	1	
1282	750383411068534784	https://pbs.twimg.com/media/CmnluwbXEAAqnkw.jpg	1	
855	696713835009417216	https://pbs.twimg.com/media/Cas5h-wWcAA3nAc.jpg	1	

In [14]:

```
df_tweets.sample(5)
```

Out[14]:

	tweet ID	retweet count	favorite count
0	798209839306514432	2954	11548
0	680100725817409536	1554	3891
0	864279568663928832	3266	15195
0	668221241640230912	215	537
0	677187300187611136	1033	2981

In [15]:

```
df_archive.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   tweet_id                    2356 non-null   int64  
 1   in_reply_to_status_id       78 non-null     float64
 2   in_reply_to_user_id         78 non-null     float64
 3   timestamp                   2356 non-null   object 
 4   source                      2356 non-null   object 
 5   text                        2356 non-null   object 
 6   retweeted_status_id         181 non-null    float64
 7   retweeted_status_user_id    181 non-null    float64
 8   retweeted_status_timestamp  181 non-null    object 
 9   expanded_urls               2297 non-null   object 
 10  rating_numerator            2356 non-null   int64  
 11  rating_denominator          2356 non-null   int64  
 12  name                        2356 non-null   object 
 13  doggo                       2356 non-null   object 
 14  floofer                     2356 non-null   object 
 15  pupper                      2356 non-null   object 
 16  puppo                       2356 non-null   object 
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

In [16]:

```
df_image.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   tweet_id  2075 non-null   int64  
 1   jpg_url   2075 non-null   object 
 2   img_num   2075 non-null   int64  
 3   p1        2075 non-null   object 
 4   p1_conf   2075 non-null   float64
 5   p1_dog    2075 non-null   bool   
 6   p2        2075 non-null   object 
 7   p2_conf   2075 non-null   float64
 8   p2_dog    2075 non-null   bool   
 9   p3        2075 non-null   object 
 10  p3_conf   2075 non-null   float64
 11  p3_dog    2075 non-null   bool   
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

In [17]:

```
df_tweets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2354 entries, 0 to 0
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   tweet ID        2354 non-null   object
 1   retweet count   2354 non-null   object
 2   favorite count  2354 non-null   object
dtypes: object(3)
memory usage: 73.6+ KB
```

In [18]:

```
list(df_image)
```

Out[18]:

```
['tweet_id',
 'jpg_url',
 'img_num',
 'p1',
 'p1_conf',
 'p1_dog',
 'p2',
 'p2_conf',
 'p2_dog',
 'p3',
 'p3_conf',
 'p3_dog']
```

In [19]:

```
list(df_tweets)
```

Out[19]:

```
['tweet ID', 'retweet count', 'favorite count']
```

In [20]:

```
list(df_archive)
```

Out[20]:

```
['tweet_id',
 'in_reply_to_status_id',
 'in_reply_to_user_id',
 'timestamp',
 'source',
 'text',
 'retweeted_status_id',
 'retweeted_status_user_id',
 'retweeted_status_timestamp',
 'expanded_urls',
 'rating_numerator',
 'rating_denominator',
 'name',
 'doggo',
 'floofer',
 'pupper',
 'puppo']
```

In [21]:

```
all_columns = pd.Series(list(df_image) + list(df_tweets) + list(df_archive)
all_columns[all_columns.duplicated()]
```

Out[21]:

```
15    tweet_id
dtype: object
```

In [22]:

```
df_archive.describe()
```

Out[22]:

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

In [23]:

```
df_image.describe()
```

Out[23]:

	tweet_id	img_num	p1_conf	p2_conf	p3_conf
count	2.075000e+03	2075.000000	2075.000000	2.075000e+03	2.075000e+03
mean	7.384514e+17	1.203855	0.594548	1.345886e-01	6.032417e-02
std	6.785203e+16	0.561875	0.271174	1.006657e-01	5.090593e-02
min	6.660209e+17	1.000000	0.044333	1.011300e-08	1.740170e-10
25%	6.764835e+17	1.000000	0.364412	5.388625e-02	1.622240e-02
50%	7.119988e+17	1.000000	0.588230	1.181810e-01	4.944380e-02
75%	7.932034e+17	1.000000	0.843855	1.955655e-01	9.180755e-02
max	8.924206e+17	4.000000	1.000000	4.880140e-01	2.734190e-01

In [24]:

```
df_tweets.describe()
```

Out[24]:

	tweet ID	retweet count	favorite count
count	2354	2354	2354
unique	2354	1724	2007
top	713919462244790272	1972	0
freq	1	5	179

In [25]:

```
df_archive.name.value_counts()
```

Out[25]:

```
None       745
a           55
Charlie     12
Lucy        11
Oliver      11
          ...
Murphy       1
Biden        1
Ben          1
Lassie       1
Jett         1
Name: name, Length: 957, dtype: int64
```

In [26]:

```
df_image.p1.value_counts()
```

Out[26]:

```
golden_retriever      150
Labrador_retriever    100
Pembroke               89
Chihuahua              83
pug                    57
                     ... 
bow                     1
pedestal                1
sundial                 1
African_grey            1
cougar                  1
Name: p1, Length: 378, dtype: int64
```

In [27]:

```
list(df_archive.text)[:5]
```

Out[27]:

```
["This is Phineas. He's a mystical boy. Only ever appears in the hole of a d
onut. 13/10 https://t.co/MgUWQ76dJU",
 "This is Tilly. She's just checking pup on you. Hopes you're doing ok. If n
ot, she's available for pats, snugs, boops, the whole bit. 13/10 https://t.c
o/0Xxu71qeIV",
 'This is Archie. He is a rare Norwegian Pouncing Corgo. Lives in the tall g
rass. You never know when one may strike. 12/10 https://t.co/wUnZnhtVJB',
 'This is Darla. She commenced a snooze mid meal. 13/10 happens to the best
of us https://t.co/tD36da7qLQ',
 'This is Franklin. He would like you to stop calling him "cute." He is a ve
ry fierce shark and should be respected as such. 12/10 #BarkWeek https://t.c
o/AtUZn91f7f']
```

In [28]:

```
df_archive.timestamp
```

Out[28]:

```
0       2017-08-01 16:23:56 +0000
1       2017-08-01 00:17:27 +0000
2       2017-07-31 00:18:03 +0000
3       2017-07-30 15:58:51 +0000
4       2017-07-29 16:00:24 +0000
                  ...            
2351    2015-11-16 00:24:50 +0000
2352    2015-11-16 00:04:52 +0000
2353    2015-11-15 23:21:54 +0000
2354    2015-11-15 23:05:30 +0000
2355    2015-11-15 22:32:08 +0000
Name: timestamp, Length: 2356, dtype: object
```

2.2 Observations

Quality issues

#	Dataframe	Issue
1	df_archive	Contains duplicate tweets (ie. retweets, as evidence of 181 count for `retweeted_status_id`)
2	df_archive	Unnecessary columns (`in_reply_to_status_id` , `in_reply_to_user_id` , `retweeted_status_id` , `source` , `retweeted_status_id` , `retweeted_status_user_id` , `retweeted_status_timestamp`)
3	df_archive	Different `tweet_id` count from `df_image` (suggests some tweets in `df_archive` do not have images)
4	df_archive	`name` column contains name 'None'

#	Dataframe	Issue
5	df_archive	`name` column contains entries ‘a’ and ‘quite’ (i.e. non-names that start with lower-case)
6	df_archive	`text` column contains hyperlink info (starting with ‘https’)
7	df_archive	`rating_numerator` column has values as low as 0 and as high as 1776 (typically between 10 and 15) and `rating_denominator` column has values as low as 0 and as high as 170 (typically 10)
8	df_archive	`timestamp` column is ‘object’ Dtype and ‘tweet_id' is 'int64' Dtype
9	df_tweets	All columns, despite being numbers, are ‘object’ Dtype and the shared observation is labeled `tweet ID`
10	df_image	Has multiple image predictions when only one is necessary

Tidiness issues

#	Dataframe	Issue
1	df_archive	Variables as column headers (`doggo` , `flooder` , `pepper` , `puppy`)
2	df_tweets + df_image	Share same observational unit as `df_archive` so they don't need to be separate dataframes

3. Cleaning Data

In [29]:

```
# Make copies of original pieces of data
df_archive_clean = df_archive.copy()
df_image_clean = df_image.copy()
df_tweets_clean = df_tweets.copy()
```

3.1 Quality issues

Issue #1:

- df_archive: Contains duplicate tweets (ie. retweets, as evidence of 181 count for `retweeted_status_id`)

Issue #1 - Define:

- Remove unnecessary retweets using boolean masking to select only entries that have null values (ie. that are "True") for `retweeted_status_id`

Issue #1 - Code

In [30]:

```
# Total number of tweets including retweets
df_archive_clean.shape[0]
```

Out[30]:

```
2356
```

In [31]:

```
# Number of retweets
df_archive_clean[df_archive_clean.retweeted_status_id.isnull()== False].cou
```

Out[31]:
```
181
```

In [32]:
```
# Boolean masking to filter out retweets
df_archive_clean = df_archive_clean[df_archive_clean.retweeted_status_id.is
```

Issue #1 - Test

In [33]:
```
# Total number of tweets (should be  2175 [2356 - 181])
df_archive_clean.shape[0]
```

Out[33]:
```
2175
```

In [34]:
```
#  Number of retweets (should be 0)
df_archive_clean[df_archive_clean.retweeted_status_id.isnull()== False].cou
```

Out[34]:
```
0
```

Issue #2:

- df_archive: Unnecessary columns (`in_reply_to_status_id` , `in_reply_to_user_id` , `retweeted_status_id` , `source` , `retweeted_status_id` , `retweeted_status_user_id` , `retweeted_status_timestamp`)

Issue #2 - Define:

- Drop unnecessary columns

Issue #2 - Code

In [35]:
```
list(df_archive_clean)
```

Out[35]:
```
['tweet_id',
 'in_reply_to_status_id',
 'in_reply_to_user_id',
 'timestamp',
 'source',
 'text',
 'retweeted_status_id',
 'retweeted_status_user_id',
 'retweeted_status_timestamp',
 'expanded_urls',
 'rating_numerator',
 'rating_denominator',
 'name',
 'doggo',
 'floofer',
 'pupper',
 'puppo']
```

In [36]:
```
# Drop unnecessary columns
unnecessary_columns = ["in_reply_to_status_id",
                       "in_reply_to_user_id",
                       "retweeted_status_id",
                       "source",
                       "retweeted_status_id",
                       "retweeted_status_user_id",
                       "retweeted_status_timestamp"]

df_archive_clean = df_archive_clean.drop(unnecessary_columns, axis=1)
```

Issue #2 - Test

In [37]:

```
list(df_archive_clean)
```

Out[37]:

```
['tweet_id',
 'timestamp',
 'text',
 'expanded_urls',
 'rating_numerator',
 'rating_denominator',
 'name',
 'doggo',
 'floofer',
 'pupper',
 'puppo']
```

Issue #3:

- df_archive: Different `tweet_id` count from `df_image` (suggests some tweets in `df_archive` do not have images)

Issue #3 - Define:

- Drop rows that are not common between `df_archive` and `df_image` using the `isin()` function to align the `tweet_id` count

Issue #3 - Code

In [38]:

```
string1 = df_archive_clean.tweet_id.count()
print("There are {} unique 'tweet_id' in the 'df_archive' table".format(str
```

```
There are 2175 unique 'tweet_id' in the 'df_archive' table
```

In [39]:

```
string2 = df_image_clean.tweet_id.count()
print("There are {} unique 'tweet_id' in the 'df_image' table".format(strin
```

```
There are 2075 unique 'tweet_id' in the 'df_image' table
```

In [40]:

```
string3 = df_archive_clean.tweet_id.isin(df_image_clean.tweet_id).sum()
print("There are {} unique 'tweet_id' that are common in both the 'df_archi
```

```
There are 1994 unique 'tweet_id' that are common in both the 'df_archive' an
d 'df_image' table
```

In [41]:

```
# Align df_archive_clean with df_image_clean
df_archive_clean = df_archive_clean[df_archive_clean.tweet_id.isin(df_image_

# Align df_image_clean with df_archive_clean
df_image_clean = df_image_clean[df_image_clean.tweet_id.isin(df_archive_cle
```

Issue #3 - Test

In [42]:

```
string1 = df_archive_clean.tweet_id.count()
print("There are {} unique 'tweet_id' in the 'df_archive' table".format(str
```

```
There are 1994 unique 'tweet_id' in the 'df_archive' table
```

In [43]:

```
string2 = df_image_clean.tweet_id.count()
print("There are {} unique 'tweet_id' in the 'df_image' table".format(strin
```

```
There are 1994 unique 'tweet_id' in the 'df_image' table
```

Issue #4

- df_archive: `name` column contains name ‘None’ (count: 745)

Issue #4 - Define

- Examine `name` column entries that contain "None" to confirm that they are entered correctly, and then fix entries if necessary.

Issue #4 - Code

In [44]:

```
# Isolate entries of "None" in the `name` column and subset the `name` and
none_names = df_archive_clean.query('name == "None"')[["name","text"]]
none_names.head()
```

Out[44]:

	name	text
5	None	Here we have a majestic great white breaching ...
7	None	When you watch your owner call another dog a g...
12	None	Here's a puppo that seems to be on the fence a...
24	None	You may not have known you needed to see this ...
25	None	This... is a Jubilant Antarctic House Bear. We...

In [45]:

```
string4 = none_names.count()[0]
print("There are {} entries with 'None' entered as the dog's name.".format(
```

```
There are 546 entries with 'None' entered as the dog's name.
```

In [46]:

```
# Use a custom function to extract the name of the dog and place the dog na
def dog_name_finder(df):
    """ Use a regex to extract the name of the dog and place the dog name i

    Keyword arguments:
    df -- String in 'text' column must contain the one of the following pat
    - "named *dog name*"
    - "name is *dog name*"

    """
    x = df.text.str.extract(r'(?:(?:name\sis\s)|(?:named\s))([a-zA-Z]+)')
    df['dog_name'] = x[0]
    df = df[x[0].isnull() == False]
    return df

none_names = dog_name_finder(none_names)
none_names
```

Out[46]:

	name	text	dog_name
168	None	Sorry for the lack of posts today. I came home...	Zoey
1678	None	We normally don't rate bears but this one seem...	Thea
1734	None	This pup's name is Sabertooth (parents must be...	Sabertooth
2166	None	Here we have a Gingivitis Pumpernickel named Z...	Zeus
2227	None	Here we have an Azerbaijani Buttermilk named G...	Guss
2267	None	Another topnotch dog. His name is Big Jumpy Ra...	Big
2269	None	This a Norwegian Pewterschmidt named Tickles. ...	Tickles

In [47]:

```
# Replace 'None' entries in `name` column with entries from `dog_name`
def dog_name_changer(df):
    """Replace values of `name` with those from `dog_name`."""
    df["name"] = df['dog_name'].values
    df = df.drop(columns="dog_name")
    return df

none_names = dog_name_changer(none_names)
none_names
```

Out[47]:

	name	text
168	Zoey	Sorry for the lack of posts today. I came home...
1678	Thea	We normally don't rate bears but this one seem...
1734	Sabertooth	This pup's name is Sabertooth (parents must be...
2166	Zeus	Here we have a Gingivitis Pumpernickel named Z...
2227	Guss	Here we have an Azerbaijani Buttermilk named G...
2267	Big	Another topnotch dog. His name is Big Jumpy Ra...
2269	Tickles	This a Norwegian Pewterschmidt named Tickles. ...

In [48]:

```
# Manually replace index 2267 name value to "Big Jumpy Rat"
none_names["name"].loc[[2267]] = "Big Jumpy Rat"
none_names
```

Out[48]:

	name	text
168	Zoey	Sorry for the lack of posts today. I came home...
1678	Thea	We normally don't rate bears but this one seem...
1734	Sabertooth	This pup's name is Sabertooth (parents must be...
2166	Zeus	Here we have a Gingivitis Pumpernickel named Z...
2227	Guss	Here we have an Azerbaijani Buttermilk named G...
2267	Big Jumpy Rat	Another topnotch dog. His name is Big Jumpy Ra...
2269	Tickles	This a Norwegian Pewterschmidt named Tickles. ...

Issue #4 - Test

In [49]:

```
# Reconfirm current state of `df_archive_clean`
txt = "This table currently has {} 'none' entries, {} rows, and {} columns.
string1 = df_archive_clean.query('name == "None"').count()[0]
string2 = df_archive_clean.shape
print(txt.format(string1, string2[0], string2[1]))
```

```
This table currently has 546 'none' entries, 1994 rows, and 11 columns.
```

In [50]:

```
# Save the index of new names
new_names_index = none_names.index

# Only change the values at the `new_names_index` using the values from `no
df_archive_clean.loc[new_names_index, 'name'] = none_names.loc[new_names_in
```

In [51]:

```
# Confirm update occurred without problems ('none' entries should be 7 fewe
txt = "This table currently has {} 'none' entries, {} rows, and {} columns.
string1 = df_archive_clean.query('name == "None"').count()[0]
string2 = df_archive_clean.shape
print(txt.format(string1, string2[0], string2[1]))
```

```
This table currently has 539 'none' entries, 1994 rows, and 11 columns.
```

Issue #5:

- df_archive: name column contains entries 'a' and 'quite' (ie. non-names that start with lower-case)

Issue #5 - Define

- Fix misentered names in the `name` column

Issue #5 - Code

In [52]:

```
# Examine and itemize misentered names in the 'name' column
df_archive_clean.name[df_archive_clean.name.str.match(r'(^[a-z])')].value_c
```

Out[52]:

```
a               55
the              7
an               6
one              4
very             4
quite            3
just             3
getting          2
space            1
not              1
my               1
his              1
unacceptable     1
officially       1
by               1
infuriating      1
incredibly       1
this             1
actually         1
all              1
such             1
light            1
Name: name, dtype: int64
```

In [53]:

```
# Get count of misentered entries
string = df_archive_clean.name[df_archive_clean.name.str.match(r'(^[a-z])')
print("There are {} misentered names in the `name` column".format(string))
```

```
There are 98 misentered names in the `name` column
```

In [54]:

```
# Isolate misentered entries in the `name` column and subset the `name` and
wrong_names = df_archive_clean[df_archive_clean.name.str.match(r'(^[a-z])')
wrong_names
```

Out[54]:

	name	text
22	such	I've yet to rate a Venezuelan Hover Wiener. Th...
56	a	Here is a pupper approaching maximum borkdrive...
169	quite	We only rate dogs. This is quite clearly a smo...
193	quite	Guys, we only rate dogs. This is quite clearly...
369	one	Occasionally, we're sent fantastic stories. Th...
...	...	...
2349	an	This is an odd dog. Hard on the outside but lo...
2350	a	This is a truly beautiful English Wilson Staff...
2352	a	This is a purebred Piers Morgan. Loves to Netf...
2353	a	Here is a very happy pup. Big fan of well-main...
2354	a	This is a western brown Mitsubishi terrier. Up...

98 rows × 2 columns

Note: Complete text contents of the `text` column were difficult to view in this notebook so I opted to extract the table to a spreadsheet for a closer look.

In [55]:

```
# Extract table to a spreadsheet
wrong_names.to_excel('wrong_names.xlsx')
```

Note: After examining the spreadsheet, I could identify 3 error-types within the **98 mismatched names**:

1. **22 entries** followed the pattern "named *dog name*" or "name is *dog name*"
2. **2 entries** had no pattern but did contain names
3. **74 entries** had no names in the tweet text

Cleaning method:

1. Use the custom functions defined in the previous section to fix the entries
2. Manually extract the names using the spreedsheet and map to the `name` column
3. Replace the `name` column entries with "None"

In [56]:

```
# 1. Use the custom functions defined in the previous section to fix the en
wrong_names_1 = dog_name_finder(wrong_names)
wrong_names_1
```

Out[56]:

	name	text	dog_name
852	my	This is my dog. Her name is Zoey. She knows I'...	Zoey
1853	a	This is a Sizzlin Menorah spaniel from Brookly...	Wylie
1955	a	This is a Lofted Aphrodisiac Terrier named Kip...	Kip
2034	a	This is a Tuscaloosa Alcatraz named Jacob (Yac...	Jacob
2066	a	This is a Helvetica Listerine named Rufus. Thi...	Rufus
2116	a	This is a Deciduous Trimester mix named Spork....	Spork
2125	a	This is a Rich Mahogany Seltzer named Cherokee...	Cherokee
2128	a	This is a Speckled Cauliflower Yosemite named ...	Hemry
2146	a	This is a spotted Lipitor Rumpelstiltskin name...	Alphred
2161	a	This is a Coriander Baton Rouge named Alfredo....	Alfredo
2191	a	This is a Slovakian Helter Skelter Feta named ...	Leroi
2204	an	This is an Irish Rigatoni terrier named Berta....	Berta
2218	a	This is a Birmingham Quagmire named Chuk. Love...	Chuk
2235	a	This is a Trans Siberian Kellogg named Alfonso...	Alfonso
2249	a	This is a Shotokon Macadamia mix named Cheryl....	Cheryl
2255	a	This is a rare Hungarian Pinot named Jessiga. ...	Jessiga
2264	a	This is a southwest Coriander named Klint. Hat...	Klint
2273	a	This is a northern Wahoo named Kohl. He runs t...	Kohl
2287	a	This is a Dasani Kingfisher from Maine. His na...	Daryl
2304	a	This is a curly Ticonderoga named Pepe. No fee...	Pepe
2311	a	This is a purebred Bacardi named Octaviath. Ca...	Octaviath
2314	a	This is a golden Buckminsterfullerene named Jo...	Johm

In [57]:

```
wrong_names_1.count()[0]
```

Out[57]:

```
22
```

In [58]:

```
# Replace values of `name` with those from `dog_name`
wrong_names_1 = dog_name_changer(wrong_names_1)
wrong_names_1
```

Out[58]:

	name	text
852	Zoey	This is my dog. Her name is Zoey. She knows I'...
1853	Wylie	This is a Sizzlin Menorah spaniel from Brookly...
1955	Kip	This is a Lofted Aphrodisiac Terrier named Kip...
2034	Jacob	This is a Tuscaloosa Alcatraz named Jacob (Yac...
2066	Rufus	This is a Helvetica Listerine named Rufus. Thi...
2116	Spork	This is a Deciduous Trimester mix named Spork....
2125	Cherokee	This is a Rich Mahogany Seltzer named Cherokee...
2128	Hemry	This is a Speckled Cauliflower Yosemite named ...
2146	Alphred	This is a spotted Lipitor Rumpelstiltskin name...
2161	Alfredo	This is a Coriander Baton Rouge named Alfredo....
2191	Leroi	This is a Slovakian Helter Skelter Feta named ...
2204	Berta	This is an Irish Rigatoni terrier named Berta....
2218	Chuk	This is a Birmingham Quagmire named Chuk. Love...
2235	Alfonso	This is a Trans Siberian Kellogg named Alfonso...
2249	Cheryl	This is a Shotokon Macadamia mix named Cheryl....
2255	Jessiga	This is a rare Hungarian Pinot named Jessiga. ...
2264	Klint	This is a southwest Coriander named Klint. Hat...
2273	Kohl	This is a northern Wahoo named Kohl. He runs t...
2287	Daryl	This is a Dasani Kingfisher from Maine. His na...
2304	Pepe	This is a curly Ticonderoga named Pepe. No fee...
2311	Octaviath	This is a purebred Bacardi named Octaviath. Ca...
2314	Johm	This is a golden Buckminsterfullerene named Jo...

In [59]:

```
# 2. Manually extract the names from using the spreedsheet, note their inde
wrong_names_2 = wrong_names.loc[[649,992]].drop(columns="dog_name")
list(wrong_names_2.text)
```

Out[59]:

```
['Here is a perfect example of someone who has their priorities in order. 1
3/10 for both owner and Forrest https://t.co/LRyMrU7Wfq',
 'That is Quizno. This is his beach. He does not tolerate human shenanigans
on his beach. 10/10 reclaim ur land doggo https://t.co/vdr7DaRSa7']
```

In [60]:

```
dog_names = ["Forrest","Quizno"]

wrong_names_2["name"] = dog_names
wrong_names_2
```

Out[60]:

	name	text
649	Forrest	Here is a perfect example of someone who has t...
992	Quizno	That is Quizno. This is his beach. He does not...

In [61]:

```
# 3. Replace remaining incorrect 74 `name` column entries with "None"
# Add data from steps 1 and 2 in order to isolate the step 3 data
wrong_names_3 = pd.concat([wrong_names, wrong_names_1, wrong_names_2])

# Drop duplicate data (misentered names)
wrong_names_3.drop_duplicates(subset="text", keep ='last', inplace=True)
```

In [62]:

```
# Isolate 74 incorrect names
wrong_names_3 = wrong_names_3[wrong_names_3.name.str.match(r'(^[a-z])')][["
wrong_names_3
```

Out[62]:

	name	text
22	such	I've yet to rate a Venezuelan Hover Wiener. Th...
56	a	Here is a pupper approaching maximum borkdrive...
169	quite	We only rate dogs. This is quite clearly a smo...
193	quite	Guys, we only rate dogs. This is quite clearly...
369	one	Occasionally, we're sent fantastic stories. Th...
...	...	...
2349	an	This is an odd dog. Hard on the outside but lo...
2350	a	This is a truly beautiful English Wilson Staff...
2352	a	This is a purebred Piers Morgan. Loves to Netf...
2353	a	Here is a very happy pup. Big fan of well-main...
2354	a	This is a western brown Mitsubishi terrier. Up...

74 rows × 2 columns

In [63]:

```
# Replace incorrect entries with "None"
wrong_names_3["name"] = "None"
wrong_names_3
```

Out[63]:

	name	text
22	None	I've yet to rate a Venezuelan Hover Wiener. Th...
56	None	Here is a pupper approaching maximum borkdrive...
169	None	We only rate dogs. This is quite clearly a smo...
193	None	Guys, we only rate dogs. This is quite clearly...
369	None	Occasionally, we're sent fantastic stories. Th...
...	...	...
2349	None	This is an odd dog. Hard on the outside but lo...
2350	None	This is a truly beautiful English Wilson Staff...
2352	None	This is a purebred Piers Morgan. Loves to Netf...
2353	None	Here is a very happy pup. Big fan of well-main...
2354	None	This is a western brown Mitsubishi terrier. Up...

74 rows × 2 columns

In [64]:

```
# Create cleaned dataframe 'right names'

right_names = pd.concat([wrong_names_3, wrong_names_2, wrong_names_1])
right_names
```

Out[64]:

	name	text
22	None	I've yet to rate a Venezuelan Hover Wiener. Th...
56	None	Here is a pupper approaching maximum borkdrive...
169	None	We only rate dogs. This is quite clearly a smo...
193	None	Guys, we only rate dogs. This is quite clearly...
369	None	Occasionally, we're sent fantastic stories. Th...
...	...	...
2273	Kohl	This is a northern Wahoo named Kohl. He runs t...
2287	Daryl	This is a Dasani Kingfisher from Maine. His na...
2304	Pepe	This is a curly Ticonderoga named Pepe. No fee...
2311	Octaviath	This is a purebred Bacardi named Octaviath. Ca...
2314	Johm	This is a golden Buckminsterfullerene named Jo...

98 rows × 2 columns

Issue #5 - Test

- 98 (74 'none' + 24 'dog name') entries that were misentered as lowercase words were cleaned.
- Out of those, 74 became 'None', so I expect an **additional 74 'None' entries** to be in the `name` column.
- The remaining 24 cleaned entries will replace the remaining misentered, so I expect **0 misentered names** to be in the `name` column.

In [65]:

```
# Reconfirm current state of `df_archive_clean`
txt = "This table currently has {} 'none' entries, {} rows, and {} columns.
string1 = df_archive_clean.query('name == "None"').count()[0]
string2 = df_archive_clean.shape
print(txt.format(string1, string2[0], string2[1]))
```

```
This table currently has 539 'none' entries, 1994 rows, and 11 columns.
```

In [66]:

```
# Reconfirm count of misentered entries
string = df_archive_clean.name[df_archive_clean.name.str.match(r'(^[a-z])')
print("There are {} misentered names in the `name` column".format(string))
```

```
There are 98 misentered names in the `name` column
```

In [67]:

```
# Save the index of new names
new_names_index = right_names.index

# Only change the values at the new_names_index using the values from df_ar
df_archive_clean.loc[new_names_index, 'name'] = right_names.loc[new_names_i
```

In [68]:
```
# Confirm update occurred without problems ('none' entries should be 74 mor

txt = "This table currently has {} 'none' entries, {} rows, and {} columns.
string1 = df_archive_clean.query('name == "None"').count()[0]
string2 = df_archive_clean.shape
print(txt.format(string1, string2[0], string2[1]))
```

```
This table currently has 613 'none' entries, 1994 rows, and 11 columns.
```

In [69]:
```
# Confirm count of misentered entries (should be 0)

string = df_archive_clean.name[df_archive_clean.name.str.match(r'(^[a-z])')
print("There are {} misentered names in the `name` column".format(string))
```

```
There are 0 misentered names in the `name` column
```

Issue #6

- df_archive: `text` column contains hyperlink info (starting with ‘https’)

Issue #6 - Define:

- Remove hyperlink data from `text` column in the `df_archive` dataframe using regex and string splitting.

Issue #6 - Code

In [70]:
```
# Check if 'text' columns contain URLs (should be "True")
string = df_archive_clean.text.str.contains(r'\shttps.+$').describe()[2]
print("Does the `text` column contain URLs that should be removed?: {}".for
```

```
Does the `text` column contain URLs that should be removed?: True
```

In [71]:
```
# Precleaning check
list(df_archive_clean.text)[:5]
```

Out[71]:
```
["This is Phineas. He's a mystical boy. Only ever appears in the hole of a d
onut. 13/10 https://t.co/MgUWQ76dJU",
 "This is Tilly. She's just checking pup on you. Hopes you're doing ok. If n
ot, she's available for pats, snugs, boops, the whole bit. 13/10 https://t.c
o/0Xxu71qeIV",
 'This is Archie. He is a rare Norwegian Pouncing Corgo. Lives in the tall g
rass. You never know when one may strike. 12/10 https://t.co/wUnZnhtVJB',
 'This is Darla. She commenced a snooze mid meal. 13/10 happens to the best
of us https://t.co/tD36da7qLQ',
 'This is Franklin. He would like you to stop calling him "cute." He is a ve
ry fierce shark and should be respected as such. 12/10 #BarkWeek https://t.c
o/AtUZn91f7f']
```

In [72]:
```
# Remove URLs
df_archive_clean["text"] = df_archive_clean.text.str.split(r'\shttps.+$', e
```

Issue #6 - Test

In [73]:
```
# Check if 'text' columns contain URLs (should be "False")
string = df_archive_clean.text.str.contains(r'\shttps.+$').describe()[2]
print("Does the `text` column contain URLs that should be removed?: {}".for
Does the `text` column contain URLs that should be removed?: False
```

In [74]:

```
# Postcleaning check
list(df_archive_clean.text)[:5]
```

Out[74]:

```
["This is Phineas. He's a mystical boy. Only ever appears in the hole of a d
onut. 13/10",
 "This is Tilly. She's just checking pup on you. Hopes you're doing ok. If n
ot, she's available for pats, snugs, boops, the whole bit. 13/10",
 'This is Archie. He is a rare Norwegian Pouncing Corgo. Lives in the tall g
rass. You never know when one may strike. 12/10',
 'This is Darla. She commenced a snooze mid meal. 13/10 happens to the best
of us',
 'This is Franklin. He would like you to stop calling him "cute." He is a ve
ry fierce shark and should be respected as such. 12/10 #BarkWeek']
```

Issue #7:

- df_archive: `rating_denominator` column has values as low as 0 and as high as 170 (typically 10)
- df_archive: `rating_numerator` column has values as low as 0 and as high as 1776 (typically between 10 and 15)

Issue #7 - Define:

a. For entries with irregular denominators (i.e. not 10), normalize both the numerator and denominator to a standard denominator of 10 \ b. For entries with irregular numerators (i.e. outliers outside of the 95th percentile but have denominators of 10), either normalize the entries using the overall median or fix an error

Issue #7a - Code

In [75]:

```
# Precleaning check
df_archive_clean[["rating_numerator", "rating_denominator"]].describe()
```

Out[75]:

	rating_numerator	rating_denominator
count	1994.000000	1994.000000
mean	12.280843	10.532096
std	41.497718	7.320710
min	0.000000	2.000000
25%	10.000000	10.000000
50%	11.000000	10.000000
75%	12.000000	10.000000
max	1776.000000	170.000000

In [76]:

```
median = df_archive_clean["rating_numerator"].median()
print("Median: {}".format(median))
```

```
Median: 11.0
```

In [77]:

```
txt = "{} entries out of 1994 correctly have a denominator of 10 so {} entr
```

```
string1 = df_archive_clean.query('rating_denominator==10').count()[0]
string2 = df_archive_clean.query('rating_denominator!=10').count()[0]
print(txt.format(string1, string2))
```

```
1976 entries out of 1994 correctly have a denominator of 10 so 18 entries ha
ve irregular denominators.
```

In [78]:

```
# Isolate entries with irregular denominators and subset only the relevant
irr_denominator = df_archive_clean.query('rating_denominator!=10')[["name",
irr_denominator
```

Out[78]:

	name	text	rating_numerator	rating_denominator
433	None	The floofs have been released I repeat the flo...	84	70
516	Sam	Meet Sam. She smiles 24/7 &amp; secretly aspir...	24	7
902	None	Why does this never happen at my front door......	165	150
1068	None	After so many requests, this is Bretagne. She ...	9	11
1120	None	Say hello to this unbelievably well behaved sq...	204	170
1165	None	Happy 4/20 from the squad! 13/10 for all	4	20
1202	Bluebert	This is Bluebert. He just saw that both #Final...	50	50
1228	None	Happy Saturday here's 9 puppers on a bench. 99...	99	90
1254	None	Here's a brigade of puppers. All look very pre...	80	80
1274	None	From left to right:\nCletus, Jerome, Alejandro...	45	50
1351	None	Here is a whole flock of puppers. 60/50 I'll ...	60	50
1433	None	Happy Wednesday here's a bucket of pups. 44/40...	44	40
1634	None	Two sneaky puppers were not initially seen, mo...	143	130
1635	None	Someone help the girl is being mugged. Several...	121	110
1662	Darrel	This is Darrel. He just robbed a 7/11 and is i...	7	11
1779	None	IT'S PUPPERGEDDON. Total of 144/120 ...I think	144	120
1843	None	Here we have an entire platoon of puppers. Tot...	88	80
2335	None	This is an Albanian 3 1/2 legged Episcopalian...	1	2

Note: After looking quickly at the text of the irregular entries, I realized that simply normalizing all 18 of these entries would be incorrect. Some are numbers were simply misindentifed as a rating, whereas they were actually just dates like 4/20 or expressions like 24/7. So I'll examine this more closely in a spreadsheet as with a previous task.

In [79]:

```
# Extract table to a spreadsheet
irr_denominator.to_excel('irr_denominator.xlsx')
```

Note: After examining the spreadsheet, I could identify 3 error-types within the **18 entries with irregular denominators**:

1. **12 entries** simply had irregular ratings
2. **5 entries** had ratings but
3. **1 entry** had no rating at all

Cleaning method: ** Here it made more sense to just quickly clean the entries within Excel and then reload the spreadsheet

1. Use a simple cross-multiplication method (numerator*10/denominator) to normalize the numerator then flash fill the other entries into new columns
2. Manually enter the ratings into new columns the correctorthe names using the spreedsheet and map to the `name` column
3. Manually enter the rating using the median of 11 (current mean of 12 was skewed by an outlier that will be corrected)

In [80]:

```
# Read corrected spreadsheet to a dataframe
irr_denominator = pd.read_excel('irr_denominator_ok.xlsx', index_col=0)
irr_denominator
```

Out[80]:

	name	text	rating_numerator	rating_denominator	new_rating_numera
433	None	The floofs have been released I repeat the flo...	84	70	
516	Sam	Meet Sam. She smiles 24/7 &amp; secretly aspir...	24	7	
902	None	Why does this never happen at my front door......	165	150	
1068	None	After so many requests, this is Bretagne. She ...	9	11	
1120	None	Say hello to this unbelievably well behaved sq...	204	170	
1165	None	Happy 4/20 from the squad! 13/10 for all	4	20	
1202	Bluebert	This is Bluebert. He just saw that both #Final...	50	50	
1228	None	Happy Saturday here's 9 puppers on a bench. 99...	99	90	
1254	None	Here's a brigade of puppers. All look very pre...	80	80	
1274	None	From left to right:\nCletus, Jerome, Alejandro...	45	50	
1351	None	Here is a whole flock of puppers. 60/50 I'll ...	60	50	
1433	None	Happy Wednesday here's a bucket of pups. 44/40...	44	40	
1634	None	Two sneaky puppers were not initially seen, mo...	143	130	
1635	None	Someone help the girl is being mugged. Several...	121	110	
1662	Darrel	This is Darrel. He just robbed a 7/11 and is i...	7	11	
1779	None	IT'S PUPPERGEDDON. Total of 144/120 ...I think	144	120	

	name	text	rating_numerator	rating_denominator	new_rating_numera
1843	None	Here we have an entire platoon of puppers. Tot...	88	80	
2335	None	This is an Albanian 3 1/2 legged Episcopalian...	1	2	

In [81]:

```
# Save the index of new ratings
new_rating_index = irr_denominator.index

# Only change the values at the new_rating_index using the values from irr_
df_archive_clean.loc[new_rating_index, 'rating_numerator'] = irr_denominato
df_archive_clean.loc[new_rating_index, 'rating_denominator'] = irr_denomina
```

Issue #7a - Test

In [82]:

```
df_archive_clean.loc[new_rating_index, ['rating_numerator','rating_denomina
```

Out[82]:

	rating_numerator	rating_denominator
433	12	10
516	11	10
902	11	10
1068	14	10
1120	12	10
1165	13	10
1202	11	10
1228	11	10
1254	10	10
1274	9	10
1351	12	10
1433	11	10
1634	11	10
1635	11	10
1662	10	10
1779	12	10
1843	11	10
2335	9	10

In [83]:

```
txt = "{} entries out of 1994 correctly have a denominator of 10 so {} entr
string1 = df_archive_clean.query('rating_denominator==10').count()[0]
string2 = df_archive_clean.query('rating_denominator!=10').count()[0]
print(txt.format(string1, string2))
```

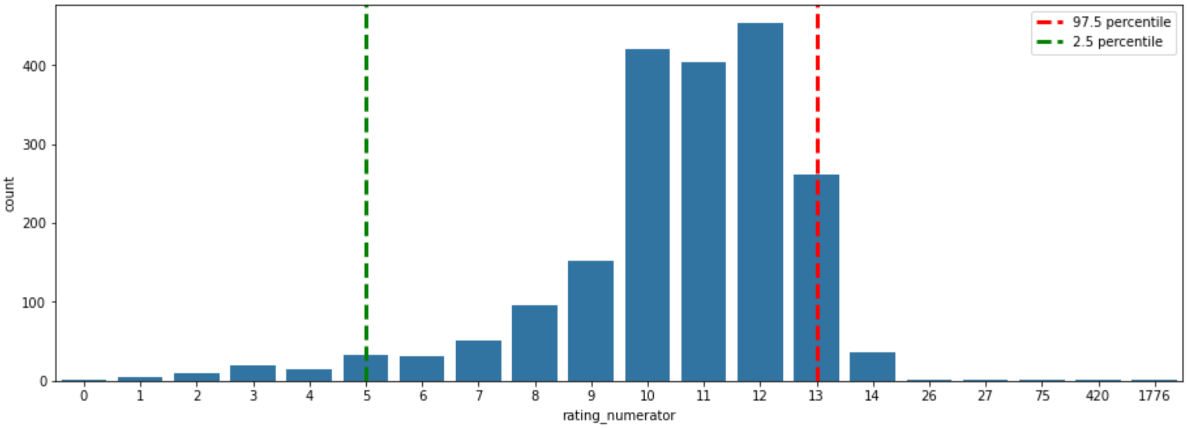
```
1994 entries out of 1994 correctly have a denominator of 10 so 0 entries hav
e irregular denominators.
```

Issue #7b - Code

In [84]:

```
# Identify upper and lower bounds for assumed outliers
lower = df_archive_clean["rating_numerator"].quantile(0.025)
upper = df_archive_clean["rating_numerator"].quantile(0.975)
txt= "95% of the tweets have ratings between {} and {}."
print(txt.format(lower,upper))

# Create a countplot to visually explore the distribution of irregular nume
fig= plt.figure()
ax = fig.add_axes([.125, .125, 2, 1])
base_color = sns.color_palette()[0]
sns.countplot(data = df_archive_clean, x = "rating_numerator", color = base_

# Add vertical bars to show upper and lower bounds
ax.axvline(x = upper, label= "97.5 percentile", color = "r", linestyle='--'
ax.axvline(x = lower, label= "2.5 percentile", color = "g", linestyle='--',

ax.legend()
plt.show()
```

```
95% of the tweets have ratings between 5.0 and 13.0.
```



In [85]:

```
txt = "{} ratings out of 1994 are outliers on the bottom low-end and {} are
string1 = df_archive_clean.query('rating_numerator < 5').count()[0]
string2 = df_archive_clean.query('rating_numerator > 13').count()[0]
print(txt.format(string1, string2))
```

```
49 ratings out of 1994 are outliers on the bottom low-end and 41 are outlier
s on the high-end.
```

In [86]:

```
# Isolate entries with irregular numerators and subset only the relevant co
irr_numerator = df_archive_clean.query('(rating_numerator<5) or (rating_num
irr_numerator = irr_numerator.sort_values(by="rating_numerator", ascending=
irr_numerator
```

Out[86]:

	name	text	rating_numerator	rating_denominator
979	Atticus	This is Atticus. He's quite simply America af....	1776	10
2074	None	After so many requests... here you go.\n\nGood...	420	10
695	Logan	This is Logan, the Chow who lived. He solemnly...	75	10
763	Sophie	This is Sophie. She's a Jubilant Bush Pupper. ...	27	10
1712	None	Here we have uncovered an entire battalion of ...	26	10
...	...	...	...	...
1869	None	What kind of person sends in a picture without...	1	10
2091	None	Flamboyant pup here. Probably poisonous. Won't...	1	10
2338	None	Not familiar with this breed. No tail (weird)....	1	10
315	None	When you're so blinded by your systematic plag...	0	10
1016	None	PUPDATE: can't see any. Even if I could, I cou...	0	10

90 rows × 4 columns

Note: Here too I'll examine this more closely in a spreadsheet.

In [87]:

```
# Extract table to a spreadsheet
irr_numerator.to_excel('irr_numerator.xlsx')
```

Note: After examining the spreadsheet, I could identify 3 issues within the **90 entries with irregular numerators**:

1. **2 entries** simply had irregular ratings
2. **3 entries** had misentered ratings because of decimal ratings like 9.75
3. **85 entries** were justified

Cleaning method: ** Here it made more sense to just quickly clean the entries within Excel and then reload the spreadsheet

1. The two entries here (1776 and 420) cannot be considered a correct rating so will change to median (11)
2. Will manually correct these
3. Will leave them as is

In [88]:

```
# Read corrected spreadsheet to a dataframe
irr_numerator = pd.read_excel('irr_numerator_ok.xlsx', index_col=0)
irr_numerator
```

Out[88]:

	name	text	rating_numerator	rating_denominator	new_rating_numerator
979	Atticus	This is Atticus. He's quite simply America af....	1776	10	11
2074	None	After so many requests... here you go.\n\nGood...	420	10	11
695	Logan	This is Logan, the Chow who lived. He solemnly...	75	10	10
763	Sophie	This is Sophie. She's a Jubilant Bush Pupper. ...	27	10	11
1712	None	Here we have uncovered an entire battalion of ...	26	10	11

In [89]:

```
# Save the index of new ratings
new_rating_index = irr_numerator.index

# Only change the values at the new_rating_index using the values from irr_
df_archive_clean.loc[new_rating_index, 'rating_numerator'] = irr_numerator.
```

Issue #7b - Test

In [90]:

```
df_archive_clean.loc[new_rating_index, ['rating_numerator','rating_denomina
```

Out[90]:

	rating_numerator	rating_denominator
979	11	10
2074	11	10
695	10	10
763	11	10
1712	11	10

In [91]:

```
# Should be 36 (41-5) outliers on the high end
txt = "{} ratings out of 1994 are outliers on the bottom low-end and {} are
string1 = df_archive_clean.query('rating_numerator < 5').count()[0]
string2 = df_archive_clean.query('rating_numerator > 13').count()[0]
print(txt.format(string1, string2))
```

```
49 ratings out of 1994 are outliers on the bottom low-end and 36 are outlier
s on the high-end.
```

In [92]:

```
# Postcleaning check
df_archive_clean[["rating_numerator", "rating_denominator"]].describe()
```

Out[92]:

	rating_numerator	rating_denominator
count	1994.000000	1994.0
mean	10.555165	10.0
std	2.176648	0.0
min	0.000000	10.0
25%	10.000000	10.0
50%	11.000000	10.0
75%	12.000000	10.0
max	14.000000	10.0

Issue #8:

- df_archive: `timestamp` column is `object` Dtype

Issue #8 - Define:

- Change dtype of `timestamp` column to `datatime` using `to_datetime`

Issue #8 - Code

In [93]:

```
# Precleaning check
df_archive_clean["timestamp"].dtypes
```

Out[93]:

```
dtype('O')
```

In [94]:

```
# Change dtype of `timestamp` column to `datatime` using `to_datetime`
df_archive_clean["timestamp"] = pd.to_datetime(df_archive_clean.timestamp)
```

Issue #8 - Test

In [95]:

```
# Postcleaning check
df_archive_clean["timestamp"].dtypes
```

Out[95]:

```
datetime64[ns, UTC]
```

Issue #9:

- All columns, despite being numbers, are 'object' Dtype and the shared observation is labeled `tweet ID` |

Issue #9 - Define:

- Change dtype of `tweet ID`, `retweet count`, and `favorite count` to `int` using the `astype` function
- Rename `tweet ID` to `tweet_id` so that it matches the naming convention of the other tables

Issue #9 - Code

In [96]:

```
# Precleaning check
df_tweets_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2354 entries, 0 to 0
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   tweet ID        2354 non-null   object
 1   retweet count   2354 non-null   object
 2   favorite count  2354 non-null   object
dtypes: object(3)
memory usage: 73.6+ KB
```

In [97]:

```
# Change dtype of all columns to `int`
df_tweets_clean[['tweet ID','retweet count','favorite count']] = df_tweets_
```

In [98]:

```
# Rename `tweet ID` to `tweet_id`
df_tweets_clean = df_tweets_clean.rename(columns = {"tweet ID":"tweet_id"})
```

Issue #9 - Test

In [99]:

```
#  Postcleaning check
df_tweets_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2354 entries, 0 to 0
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   tweet_id        2354 non-null   int64
 1   retweet count   2354 non-null   int64
 2   favorite count  2354 non-null   int64
dtypes: int64(3)
memory usage: 73.6 KB
```

Issue #10

- df_image: Has multiple image predictions when only one is necessary

Issue #10 - Define:

- Drop all columns except for `tweet_id` , `jpg_url` , and `p1`
- Rename 'p1' to 'breed'

Note: For the time being, all other image prediction data are beyond the scope of this project so should be dropped.

Issue #10 - Code

In [100…

```
#  Precleaning check
list(df_image_clean.columns)
```

Out[100]:

```
['tweet_id',
 'jpg_url',
 'img_num',
 'p1',
 'p1_conf',
 'p1_dog',
 'p2',
 'p2_conf',
 'p2_dog',
 'p3',
 'p3_conf',
 'p3_dog']
```

In [101…

```
df_image_clean.head(3)
```

Out[101]:

	tweet_id	jpg_url	img_num	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Wels
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	

In [102…

```
# Subset out unnecessary columns
df_image_clean = df_image_clean[['tweet_id', 'jpg_url', 'p1']]
```

In [103…

```
# Rename column
df_image_clean = df_image_clean.rename(columns = {"p1":"breed"})
```

Issue #10 - Test

In [104…

```
# Postcleaning check
list(df_image_clean.columns)
```

Out[104]:

```
['tweet_id', 'jpg_url', 'breed']
```

In [105…

```
df_image_clean.head(3)
```

Out[105]:

	tweet_id	jpg_url	
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	Welsh_springer_
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	re
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	German_sh

Tidiness issues

Issue #1:

- df_archive: Variables as column headers (`doggo` , `flooder` , `pepper` , `puppy`)

Issue #1 - Define:

- Extract dog stage names in `text` and, if found, add them to a new column `dog_stages`

Issue #1 - Code

In [106…

```
# Precleaning check
df_archive_clean.columns
```

Out[106]:

```
Index(['tweet_id', 'timestamp', 'text', 'expanded_urls', 'rating_numerato
r',
       'rating_denominator', 'name', 'doggo', 'floofer', 'pupper', 'pupp
o'],
      dtype='object')
```

In [107…

```
# Extract dog stage names in `text` and, if found, add them to a new column
df_archive_clean['dog_stages'] = df_archive_clean.text.str.extract('(doggo|
```

Reference: https://knowledge.udacity.com/questions/111929

In [108…

```
# Drop unnecessary columns
df_archive_clean.drop(['doggo','floofer','pupper','puppo'], axis=1, inplace
```

Issue #1 - Test

In [109…

```
# Postcleaning check
df_archive_clean.columns
```

Out[109]:

```
Index(['tweet_id', 'timestamp', 'text', 'expanded_urls', 'rating_numerato
r',
       'rating_denominator', 'name', 'dog_stages'],
      dtype='object')
```

In [110…

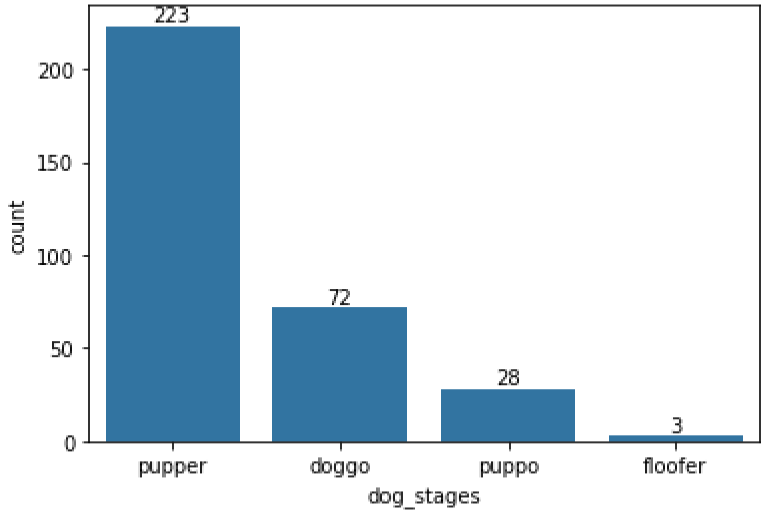
```
df_archive_clean.head(3)
```

Out[110]:

	tweet_id	timestamp	text	expand
0	892420643555336193	2017-08-01 16:23:56+00:00	This is Phineas. He's a mystical boy. Only eve...	https://twitter.com/dog_rates/status/8924
1	892177421306343426	2017-08-01 00:17:27+00:00	This is Tilly. She's just checking pup on you....	https://twitter.com/dog_rates/status/8921
2	891815181378084864	2017-07-31 00:18:03+00:00	This is Archie. He is a rare Norwegian Pouncin...	https://twitter.com/dog_rates/status/891

In [111…

```
stage_count = df_archive_clean.dog_stages.value_counts()
sns.countplot(data=df_archive_clean, x='dog_stages', order=stage_count.inde
# Print value on each bar
for i in range (stage_count.shape[0]):
    count = stage_count[i]
    plt.text(i, count+11, count, ha = 'center', va='top')
```



Issue #2:

- `df_tweets + df_image` : Share same observational unit as `df_archive` so they don't need to be separate dataframes

Issue #2 - Define:

- Merge `df_tweets_clean` to `df_archive_clean` to create `df_master`
- Merge `df_image_clean` to `df_master`

Issue #2 - Code

In [112…

```
# Merge dfs to create `df_master`

df_master = pd.merge(df_archive_clean, df_tweets_clean, how="inner", on = "
df_master = pd.merge(df_master, df_image_clean, how="inner", on = "tweet_id
df_master.reset_index(drop=True, inplace=True)
```

Issue #2 - Test

In [113…

```
# Confirm
df_master.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1994 entries, 0 to 1993
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype              
---  ------              --------------  -----              
 0   tweet_id            1994 non-null   int64              
 1   timestamp           1994 non-null   datetime64[ns, UTC]
 2   text                1994 non-null   object             
 3   expanded_urls       1994 non-null   object             
 4   rating_numerator    1994 non-null   int64              
 5   rating_denominator  1994 non-null   int64              
 6   name                1994 non-null   object             
 7   dog_stages          326 non-null    object             
 8   retweet count       1994 non-null   int64              
 9   favorite count      1994 non-null   int64              
 10  jpg_url             1994 non-null   object             
 11  breed               1994 non-null   object             
dtypes: datetime64[ns, UTC](1), int64(5), object(6)
memory usage: 187.1+ KB
```

In [114…

```
# Confirm
df_master.describe()
```

Out[114]:

	tweet_id	rating_numerator	rating_denominator	retweet count	favorite count
count	1.994000e+03	1994.000000	1994.0	1994.000000	1994.000000
mean	7.358508e+17	10.555165	10.0	2766.753260	8895.725677
std	6.747816e+16	2.176648	0.0	4674.698447	12213.193181
min	6.660209e+17	0.000000	10.0	16.000000	81.000000
25%	6.758475e+17	10.000000	10.0	624.750000	1982.000000
50%	7.084748e+17	11.000000	10.0	1359.500000	4136.000000
75%	7.877873e+17	12.000000	10.0	3220.000000	11308.000000
max	8.924206e+17	14.000000	10.0	79515.000000	132810.000000

In [115…

```
# Confirm
df_master.head(3)
```

Out[115]:

	tweet_id	timestamp	text	expand
0	892420643555336193	2017-08-01 16:23:56+00:00	This is Phineas. He's a mystical boy. Only eve...	https://twitter.com/dog_rates/status/89242
1	892177421306343426	2017-08-01 00:17:27+00:00	This is Tilly. She's just checking pup on you....	https://twitter.com/dog_rates/status/8921
2	891815181378084864	2017-07-31 00:18:03+00:00	This is Archie. He is a rare Norwegian Pouncin...	https://twitter.com/dog_rates/status/8918

4. Storing Data

Save gathered, assessed, and cleaned master dataset to a CSV file named "twitter_archive_master.csv".

In [116…

```
df_master.to_csv("twitter_archive_master.csv")
```

5. Analyzing and Visualizing Data

5.1 Insights:

1. What is the most retweeted tweet?

2. What is the most common rating?

3. What are the most common breeds found by the neural network?

In [117…

```
# 1. What is the most retweeted tweet?
most_retweeted = df_master[df_master["retweet count"] == df_master["retweet
most_retweeted
```

Out[117]:

	tweet_id	timestamp	text	expande
775	744234799360020481	2016-06-18 18:26:18+00:00	Here's a doggo realizing you can stand in a po...	https://twitter.com/dog_rates/status/74423

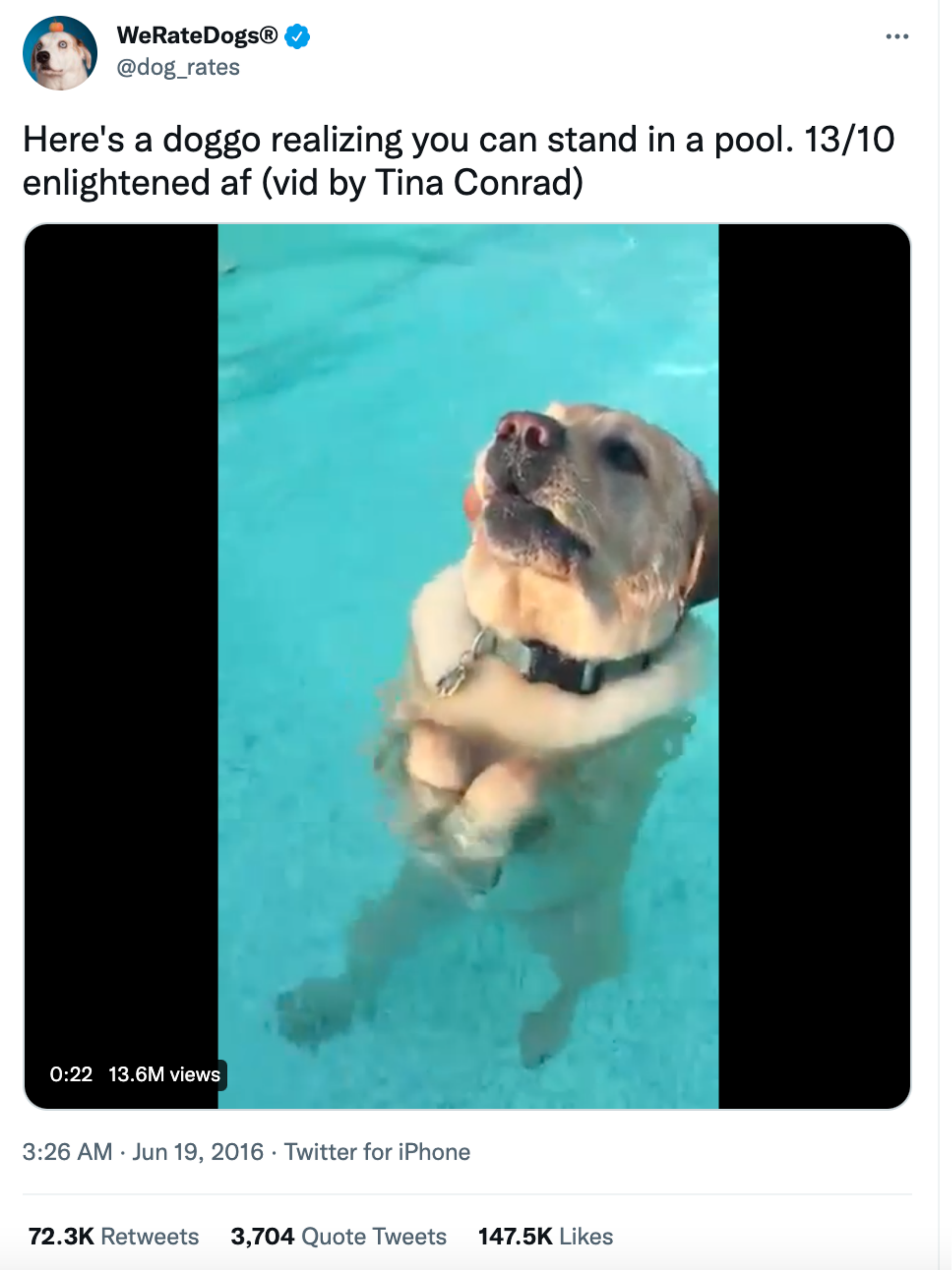
In [118…

```
list(most_retweeted["expanded_urls"])
```

Out[118]:
```
['https://twitter.com/dog_rates/status/744234799360020481/video/1']
```

In [119…
```
from IPython.display import Image
Image(filename='most_retweeted.png', width=500)
```

Out[119]:



WeRateDogs® @dog_rates

Here's a doggo realizing you can stand in a pool. 13/10 enlightened af (vid by Tina Conrad)

3:26 AM · Jun 19, 2016 · Twitter for iPhone

72.3K Retweets 3,704 Quote Tweets 147.5K Likes

In [120…
```
list(most_retweeted["jpg_url"])[0]
```

Out[120]:
```
'https://pbs.twimg.com/ext_tw_video_thumb/744234667679821824/pu/img/1GaWmtJ
tdqzZV7jy.jpg'
```

Answer: The most retweeted tweet is tweet_id `744234799360020481' which features the following very good boy.

In [121…

```
# 2. What is the most common rating?
common_rating = df_master["rating_numerator"].value_counts(normalize=True)
common_rating[:5]
```

Out[121]:

```
12    0.227683
10    0.211635
11    0.204614
13    0.131394
9     0.076229
Name: rating_numerator, dtype: float64
```

Answer: The most common rating is 12/10

In [122…

```
# 3. What are the most common breeds found by the neural network?
common_breeds = df_master["breed"].value_counts()
common_breeds[:5]
```

Out[122]:

```
golden_retriever      139
Labrador_retriever     95
Pembroke               88
Chihuahua              79
pug                    54
Name: breed, dtype: int64
```

Answer: In order, the most common breeds identified by the neural network are Golden Retriever, Labrador Retriever, Pembroke, Chihuahua, and Pug.

In [123…

```
# Code for Act_Report for image URLs for each dog breed

list(df_master[df_master["breed"] == "pug"]["jpg_url"])[0]
list(df_master[df_master["breed"] == "Chihuahua"]["jpg_url"])[0]
list(df_master[df_master["breed"] == "Pembroke"]["jpg_url"])[0]
list(df_master[df_master["breed"] == "Labrador_retriever"]["jpg_url"])[0]
list(df_master[df_master["breed"] == "golden_retriever"]["jpg_url"])[0]
```

Out[123]:

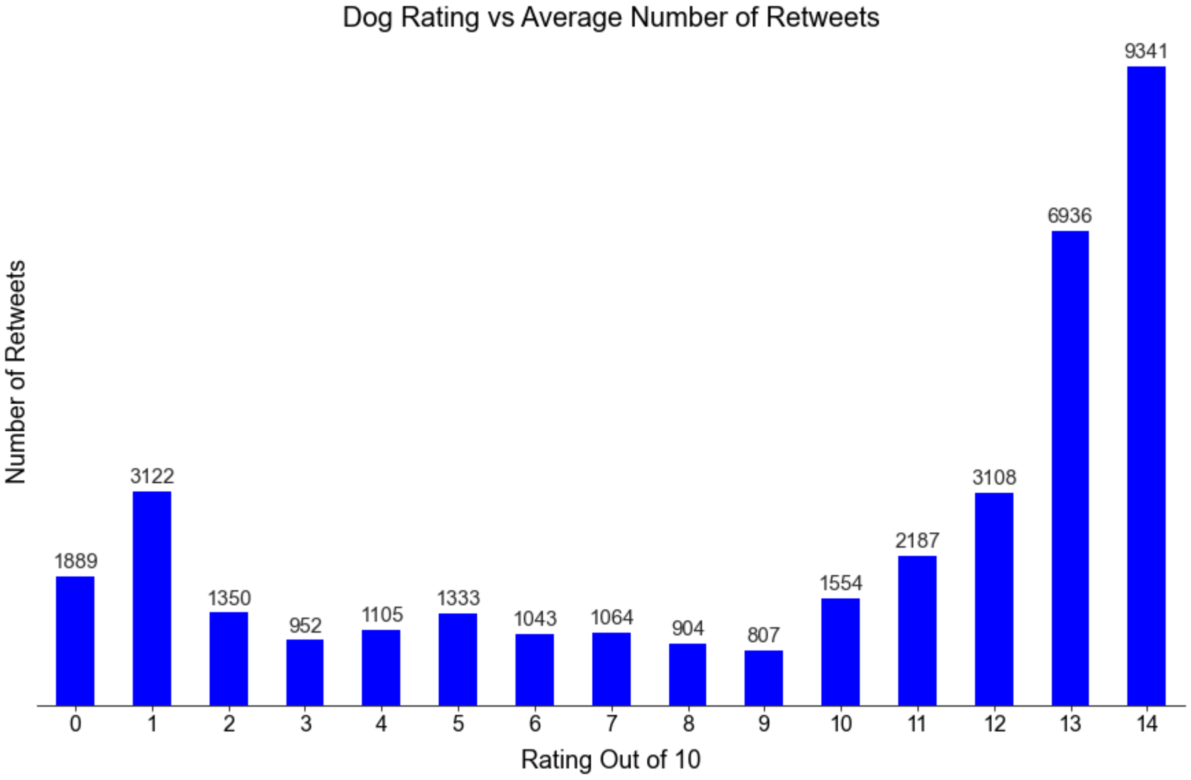
```
'https://pbs.twimg.com/media/DFg_2PVW0AEHN3p.jpg'
```

5.2 Visualization

- What is the average retweet count for each rating?

In [124…

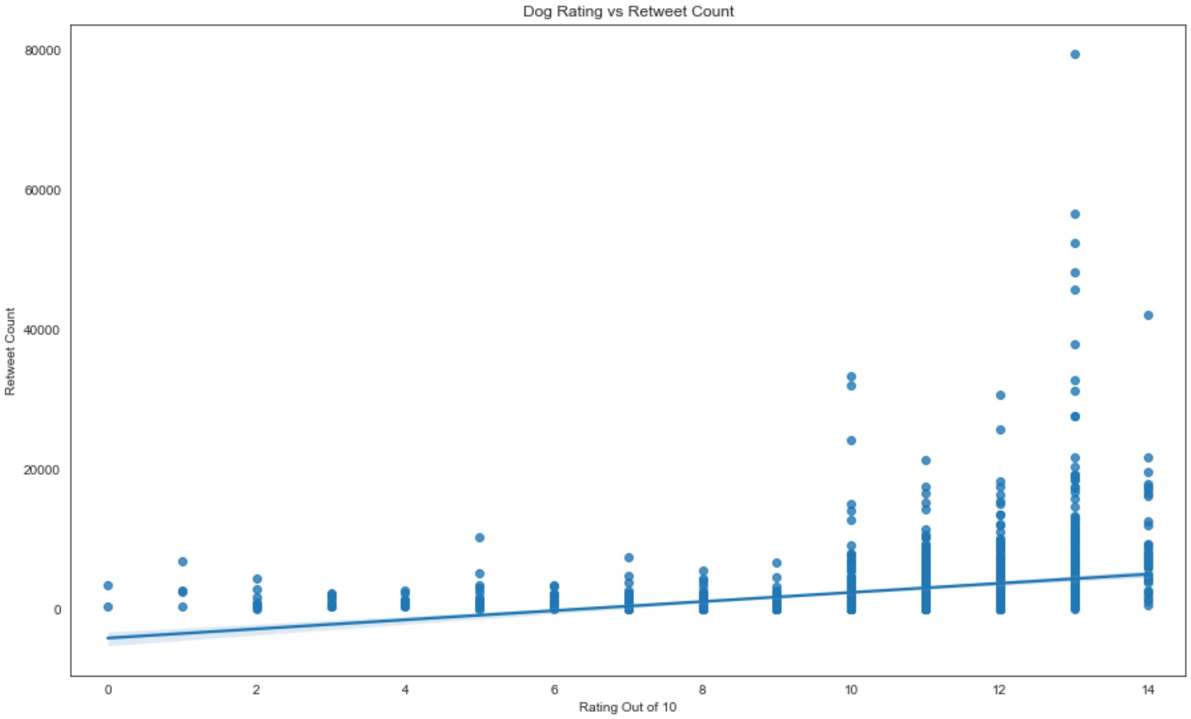
```
rating_mean = df_master.groupby("rating_numerator")["retweet count"].mean()

# Plot the results of movie_stats_total
plt.figure(figsize = (15,9))
rating_mean.plot.bar(color="b")

# Set base style
sns.set_style("white")
sns.despine(top=True,
           right=True,
           left=True)

# Print value on each bar
for i in range (rating_mean.shape[0]):
    count = int(rating_mean[i])
    plt.text(i, count+350, count, ha = 'center', va='top', size=15)

# Customize plot title and labels
plt.title("Dog Rating vs Average Number of Retweets", size=20)
plt.xlabel('Rating Out of 10', fontsize=18, labelpad= 10, color="black")
plt.ylabel('Number of Retweets', fontsize=18, color="black")
plt.xticks(fontsize=16)
plt.xticks(rotation=0, fontsize=16)
plt.yticks(ticks=[])
#plt.yticks(fontsize=16)

# Save figure
plt.savefig("Dog Rating vs Average Number of Retweets")

# Show figure
plt.show()
```



Extra, unused visualizations

In [125…

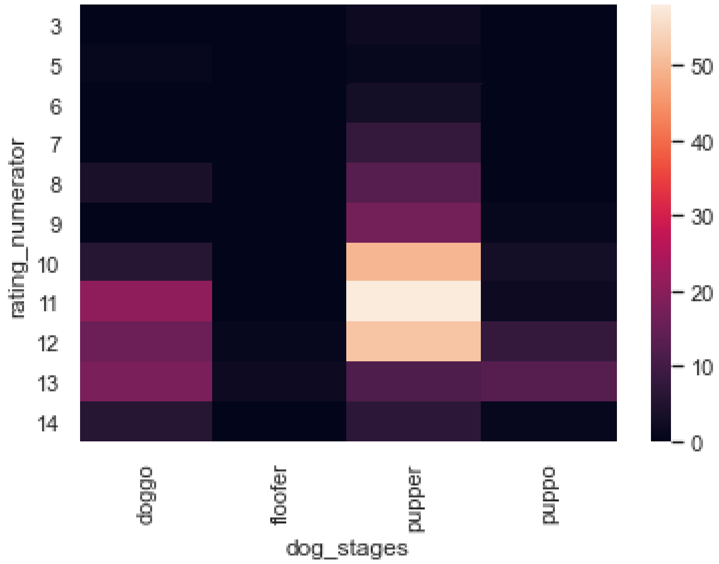
```
import seaborn as sns
import scipy.stats

fig, ax = plt.subplots(figsize=(15,9))

sns.set()

sns.regplot(x ="rating_numerator",
            y ="retweet count",
            data = df_master,
           ax = ax)

ax.set(xlabel="Rating Out of 10",
       xlim= (-0.5,14.5),
      ylabel= "Retweet Count",
      title= "Dog Rating vs Retweet Count")

plt.show()
```



In [126…

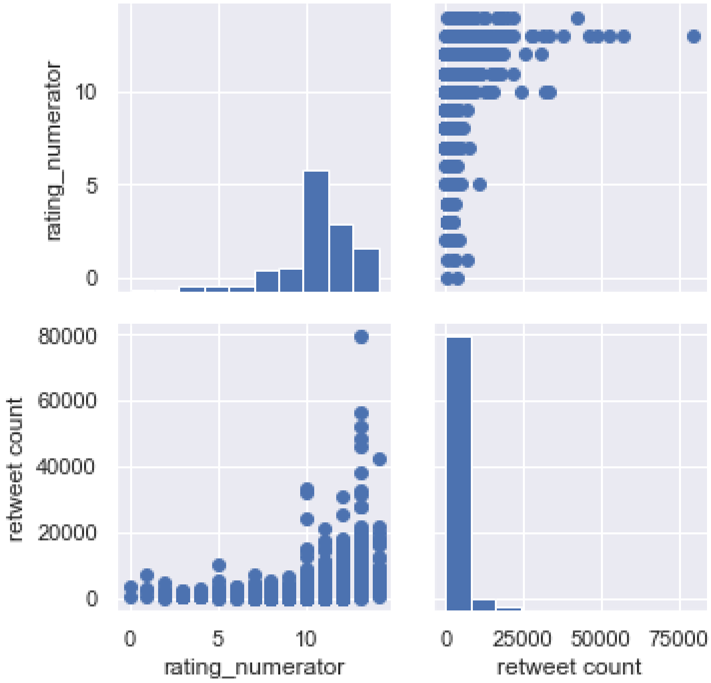
```
pd_crosstab = pd.crosstab(df_master["rating_numerator"], df_master["dog_sta
print(pd_crosstab)

# Plot a heatmap of the table
sns.heatmap(pd_crosstab)

# Rotate tick marks for visibility
plt.yticks(rotation=0)
plt.xticks(rotation=90)

plt.show()
```

```
dog_stages          doggo  floofer  pupper  puppo
rating_numerator
3                       0        0       2      0
5                       1        0       1      0
6                       0        0       3      0
7                       0        0       8      0
8                       4        0      13      0
9                       0        0      17      1
10                      6        0      50      3
11                     21        0      58      2
12                     16        1      52      8
13                     18        2      12     13
14                      6        0       7      1
```



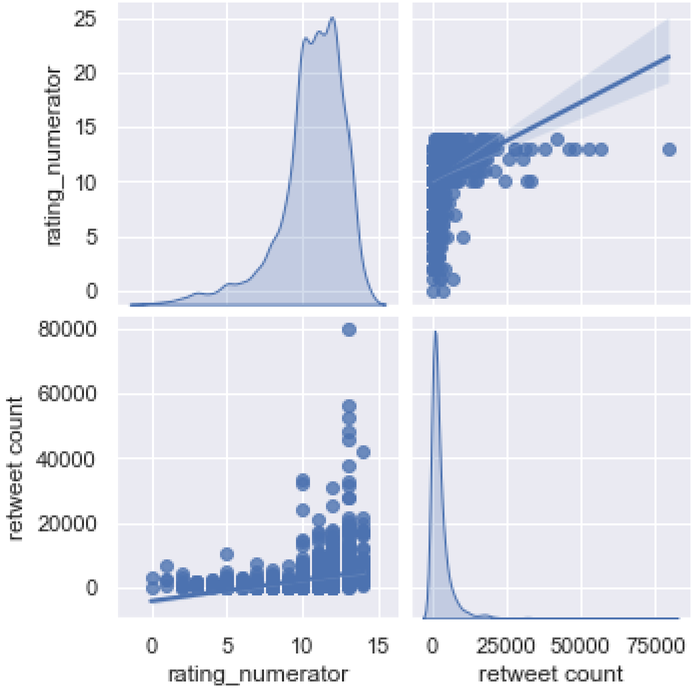
In [127…

```
g = sns.PairGrid(df_master, vars=["rating_numerator","retweet count"])
g = g.map_diag(plt.hist)
g = g.map_offdiag(plt.scatter)
```



In [128…

```
sns.pairplot(data=df_master,
             vars=["rating_numerator","retweet count"],
             kind='reg',
             palette='BrBG',
             diag_kind = 'kde')

plt.show()
plt.clf()
```



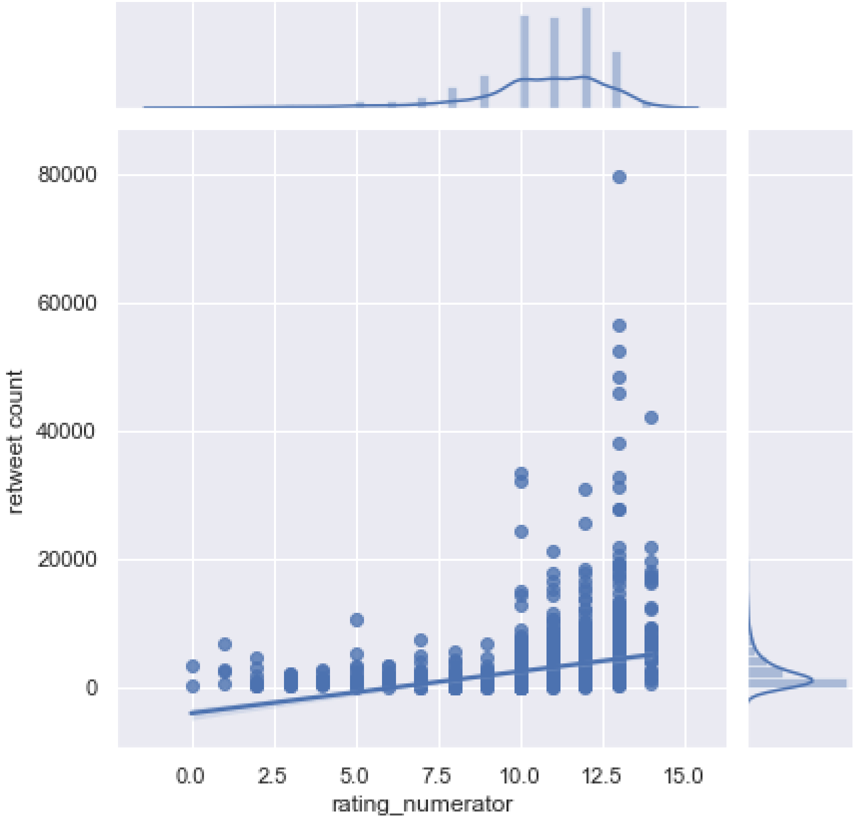
```
<Figure size 432x288 with 0 Axes>
```

In [129…

```
g = sns.JointGrid(data=df_master, x="rating_numerator",
y="retweet count")
g.plot(sns.regplot, sns.distplot)
```

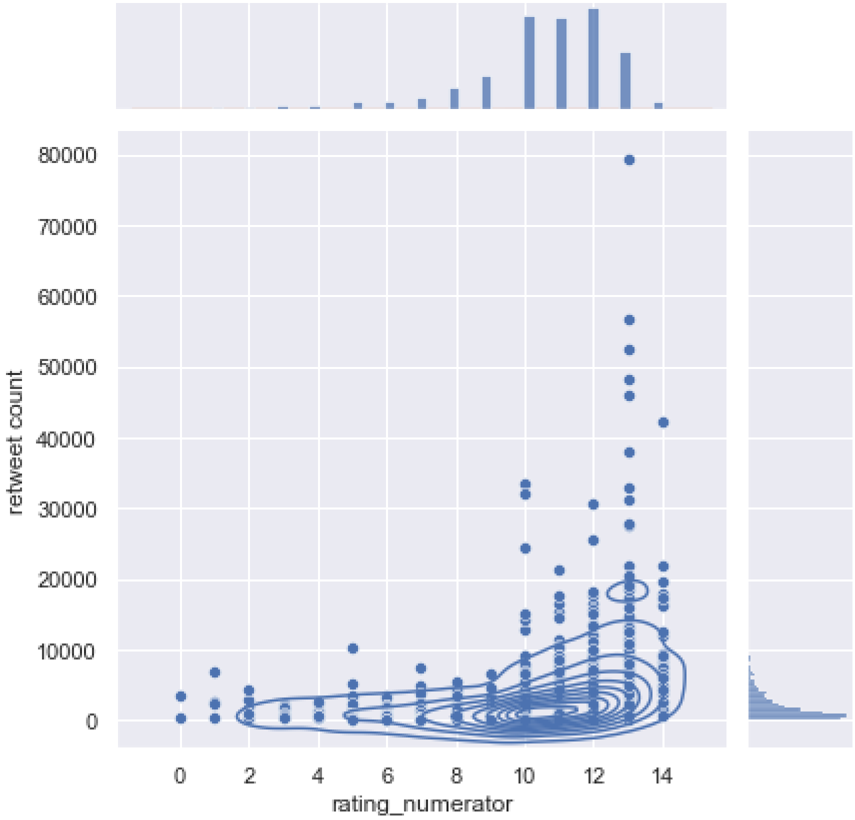
Out[129]:

```
<seaborn.axisgrid.JointGrid at 0x7f8848e4c820>
```



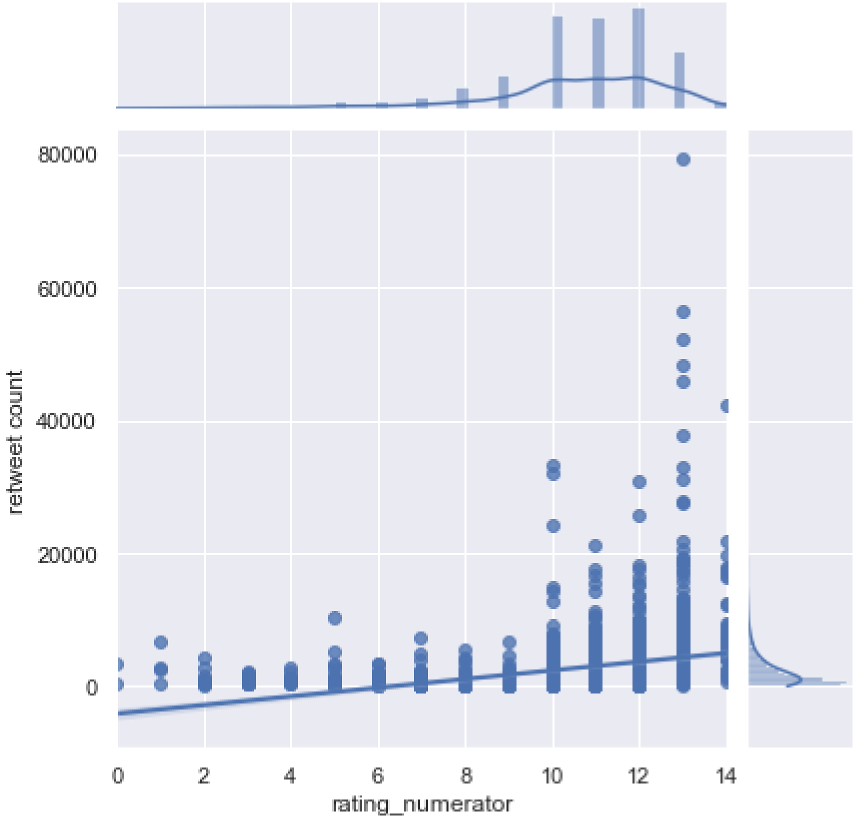
In [130…

```
import scipy.stats as stats

g = sns.jointplot(data=df_master, x="rating_numerator",
y="retweet count")
g = g.plot_joint(sns.kdeplot)
g = g.plot_marginals(sns.kdeplot, shade=True)
```



In [131…

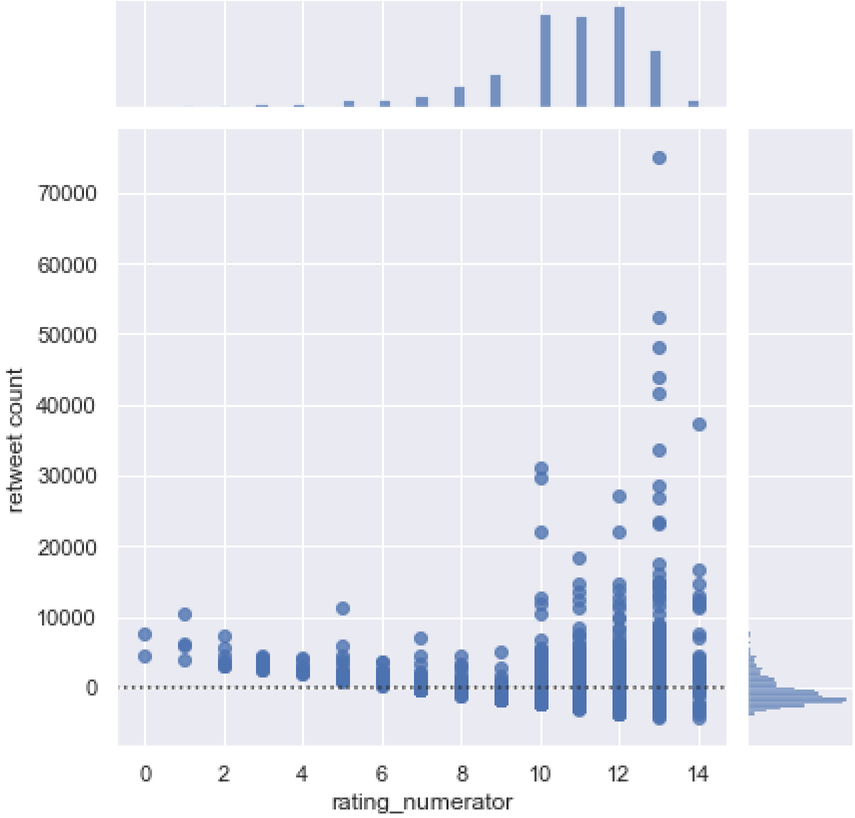
```
sns.jointplot(x="rating_numerator",
        y="retweet count",
        kind='reg',
        data=df_master)

plt.show()
plt.clf()
```



```
<Figure size 432x288 with 0 Axes>
```

In [132…

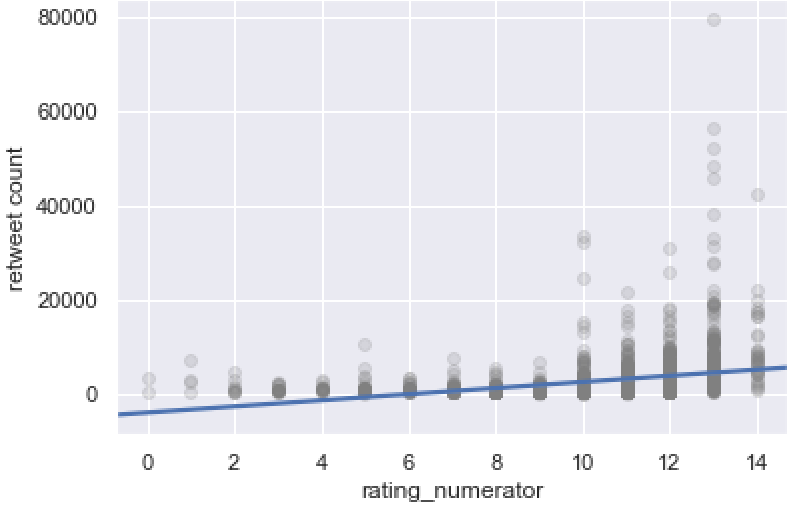
```
sns.jointplot(x="rating_numerator",
        y="retweet count",
        kind='resid',
        data=df_master)

plt.show()
plt.clf()
```



```
<Figure size 432x288 with 0 Axes>
```

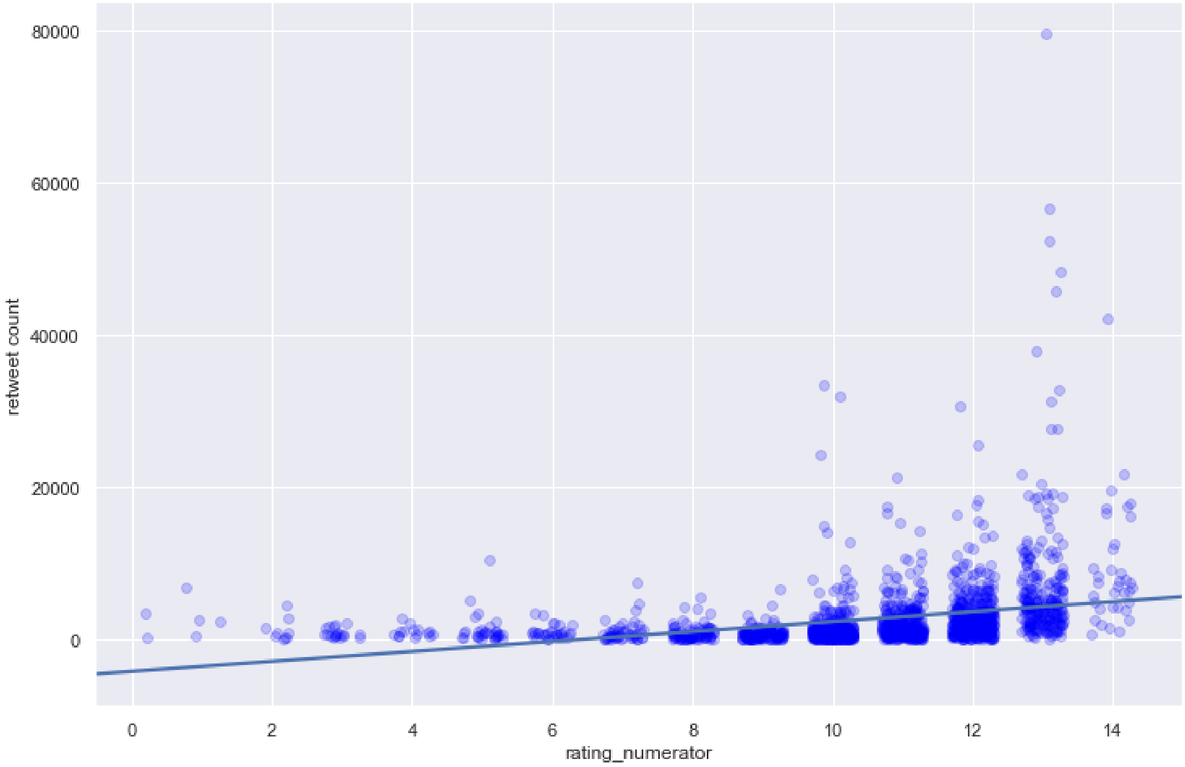
In [138…

```
sns.regplot(x = 'rating_numerator',
            y = 'retweet count',
            # Set scatter point opacity & color
            scatter_kws = {'alpha':0.2, 'color':'gray'},
            # Disable confidence band
            ci = False,
            truncate=False,
            data = df_master)

plt.show()
```



In [149…

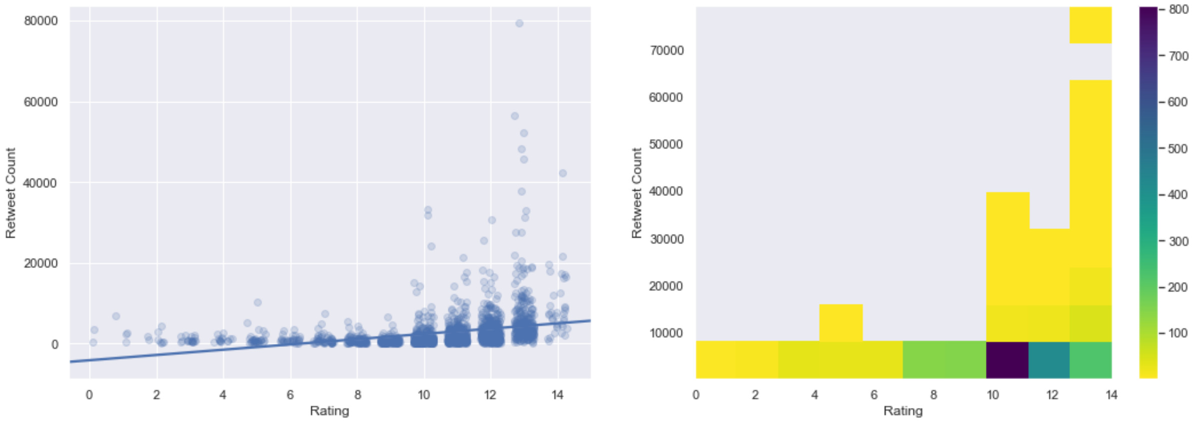
```
plt.figure(figsize=[12,8])

sns.regplot(x = 'rating_numerator',
            y = 'retweet count',
            # Set scatter point opacity & color
            scatter_kws = {'alpha':0.2, 'color':'blue'},
            # Disable confidence band
            ci = False,
            truncate=False,
            x_jitter=0.3,
            data = df_master)

plt.show()
```



In [156…

```
# Example 1. Default heat plot using Matplotlib.pyplot.hist2d() function

plt.figure(figsize=[18, 6])

# Plot on left
plt.subplot(1, 2, 1)
sns.regplot(x = 'rating_numerator',
            y = 'retweet count',
            # Set scatter point opacity & color
            scatter_kws = {'alpha':0.2},
            # Disable confidence band
            ci = False,
            truncate=False,
            x_jitter=0.3,
            data = df_master)
plt.xlabel("Rating")
plt.ylabel("Retweet Count")

# Plot on right
plt.subplot(1, 2, 2)
plt.hist2d(data = df_master,
           x = 'rating_numerator',
           y = 'retweet count', cmin=0.5, cmap='viridis_r')
```

```
plt.colorbar()
plt.xlabel("Rating")
plt.ylabel("Retweet Count");
```



In [157…

```
df_master[["rating_numerator", "retweet count"]].describe()
```

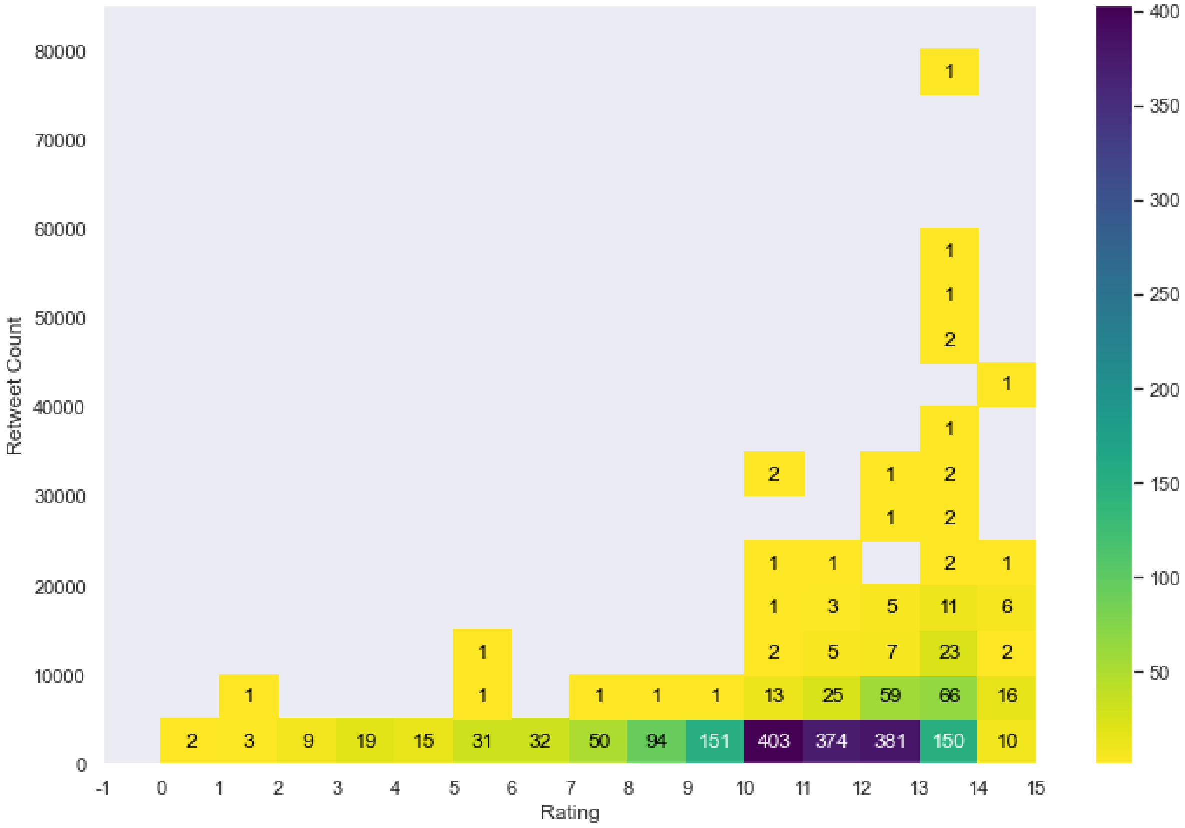
Out[157]:

```
       rating_numerator  retweet count
count       1994.000000    1994.000000
 mean         10.555165    2766.753260
  std          2.176648    4674.698447
  min          0.000000      16.000000
  25%         10.000000     624.750000
  50%         11.000000    1359.500000
  75%         12.000000    3220.000000
  max         14.000000   79515.000000
```

In [171…

```
import numpy as np
plt.figure(figsize=[12,8])
bins_x= np.arange(-1, 14+2, 1)
bins_y= np.arange(-1, 80000+5000, 5000)
ticks = bins_x
labels= ('{}'.format(x) for x in ticks)
h2d=plt.hist2d(data = df_master,
           x = 'rating_numerator',
           y = 'retweet count', cmin=0.5, cmap='viridis_r',
          bins=[bins_x, bins_y])
plt.colorbar()
plt.xlabel("Rating")
plt.ylabel("Retweet Count")
plt.xticks(ticks=bins_x, labels=labels);

# Select the bi-dimensional histogram, a 2D array of samples x and y.
# Values in x are histogrammed along the first dimension and
# values in y are histogrammed along the second dimension.
counts = h2d[0]

# Add text annotation on each cell
# Loop through the cell counts and add text annotations for each
for i in range(counts.shape[0]):
    for j in range(counts.shape[1]):
        c = counts[i,j]
        if c >= 100: # increase visibility on darker cells
```

```
            plt.text(bins_x[i]+0.5, bins_y[j]+2500, int(c),
                     ha = 'center', va = 'center', color = 'white')
        elif c > 0:
            plt.text(bins_x[i]+0.5, bins_y[j]+2500, int(c),
                     ha = 'center', va = 'center', color = 'black')
```



In [ ]: