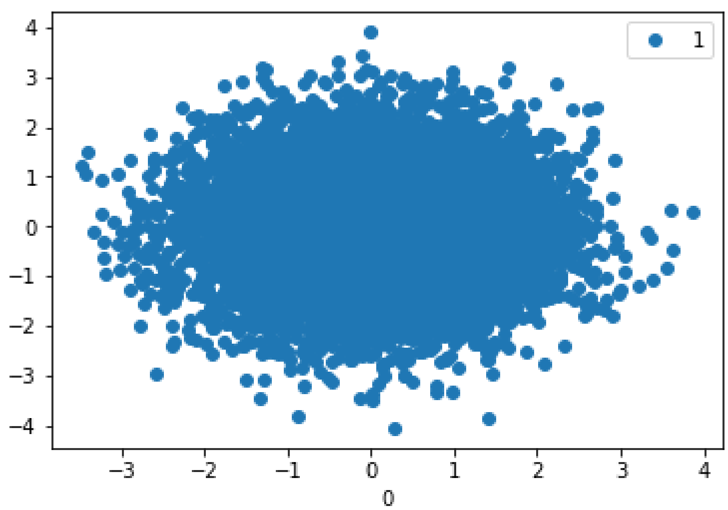
Question 1

(a) Data is frequently correlated. When the data is stacked in a matrix (which we call a data matrix), an easy way to tell if any two columns are correlated is to look at a scatter plot of each column against each other column. For a warm up, do this: Look at the data in DF1 in Lab2.zip. Which columns are (pairwise) correlated? Figure out how to do this with Pandas, and also how to do this with Seaborn.

In [1]:
```
import pandas as pd
import seaborn as sb
import numpy as np
import matplotlib.pyplot as plt
```

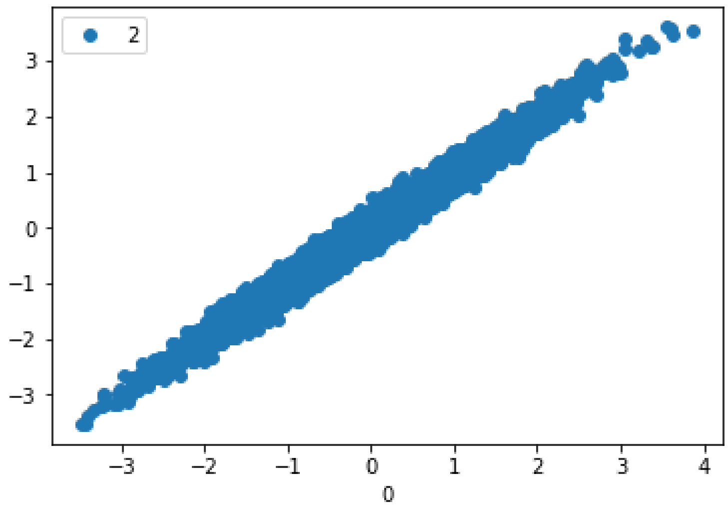
In [2]:
```
df = pd.read_csv("DF1", header=0, na_values='?', index_col=0)
df.plot(x=0, y=1, style='o')
```

Out[2]: <matplotlib.axes._subplots.AxesSubplot at 0x1a12949e48>



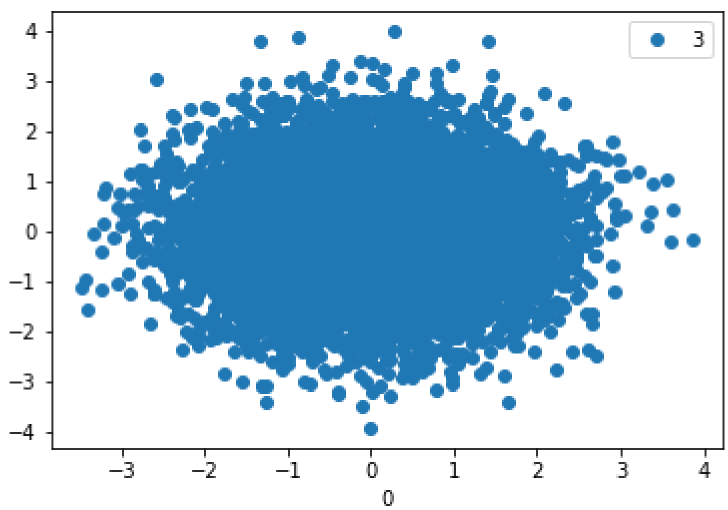
In [3]:
```
df.plot(x=0, y=2, style='o')
```

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x1a12f79cf8>



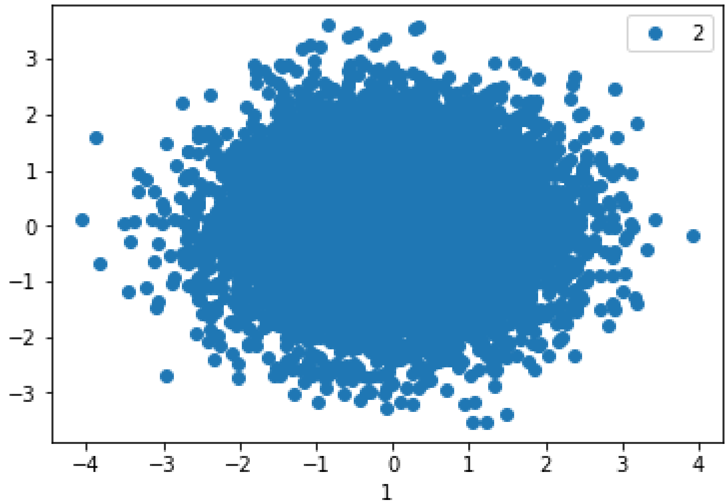
In [4]:
```
df.plot(x=0, y=3, style='o')
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x1a128daeb8>



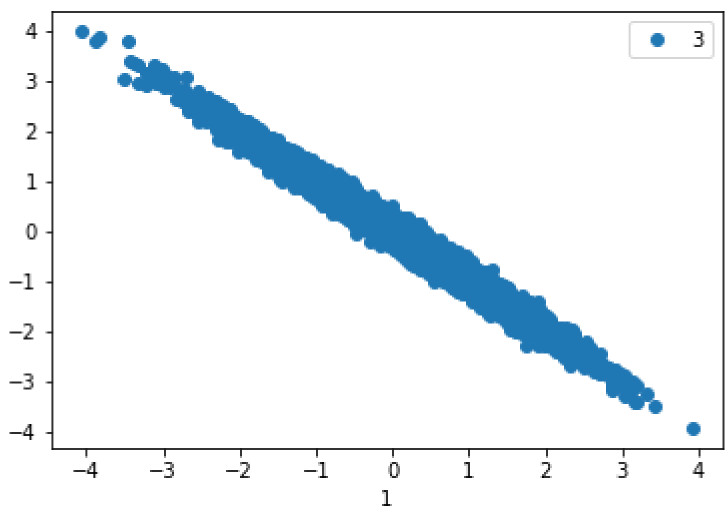
In [5]:
```
df.plot(x=1, y=2, style='o')
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1e78c7f0>



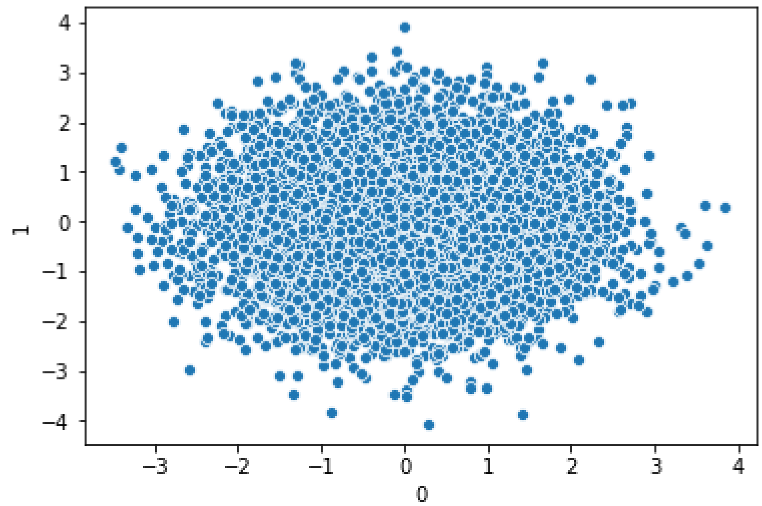
In [6]:
```
df.plot(x=1, y=3, style='o')
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1e9dbc50>



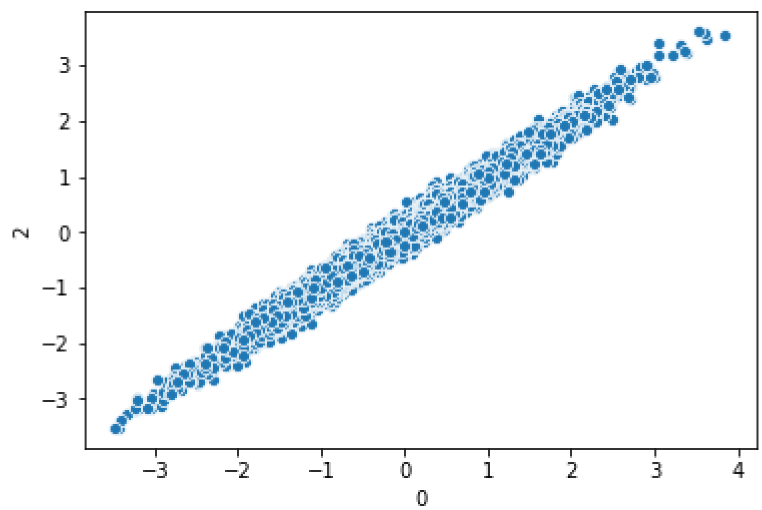
In [7]:
```
sb.scatterplot(data=df,x='0',y='1')
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1eb2a7f0>



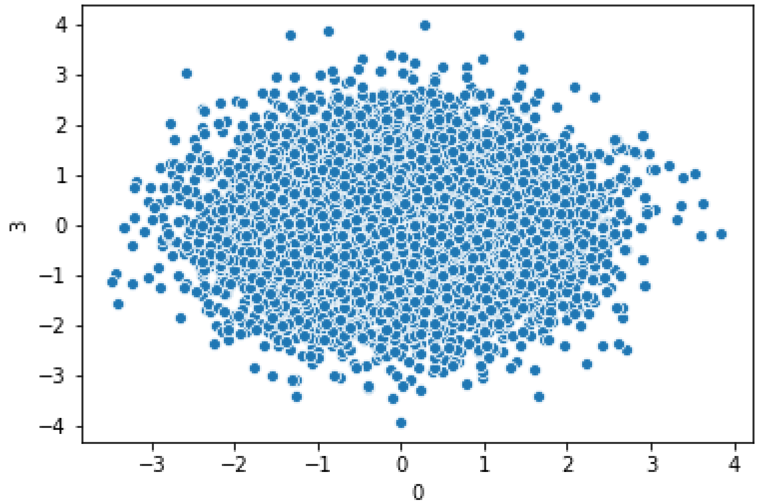
In [8]:
```
sb.scatterplot(data=df,x='0',y='2')
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1eca4ac8>



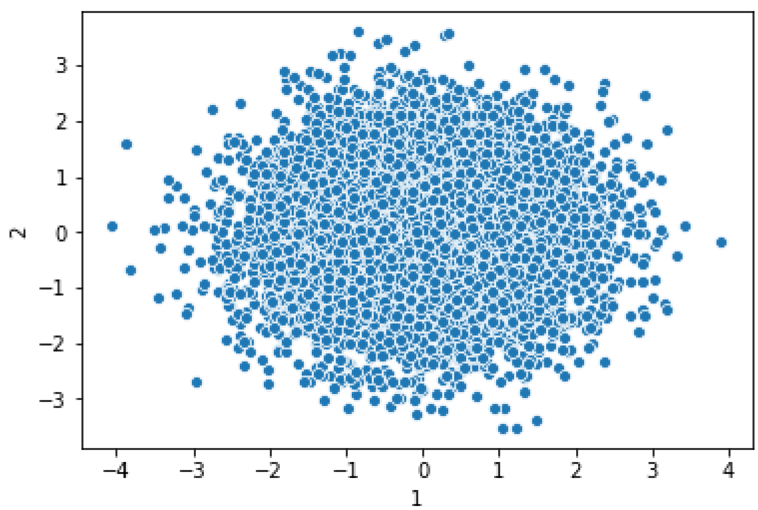
In [9]:
```
sb.scatterplot(data=df,x='0',y='3')
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1ede0048>



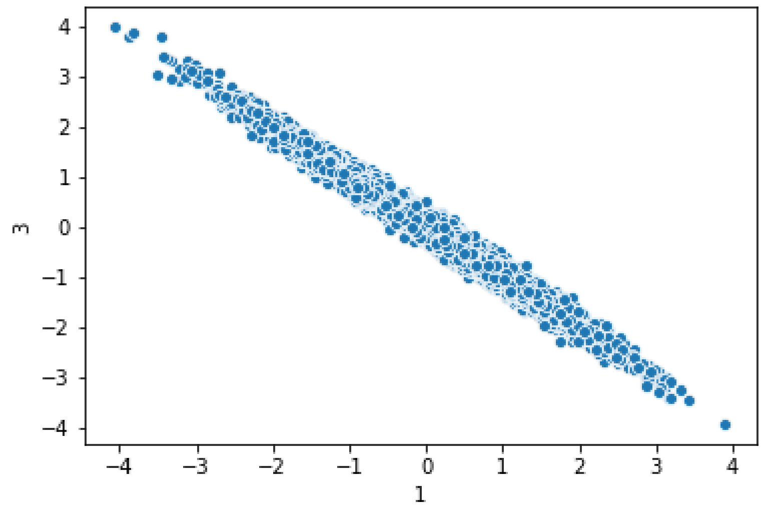
In [10]:
```
sb.scatterplot(data=df,x='1',y='2')
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1eece208>



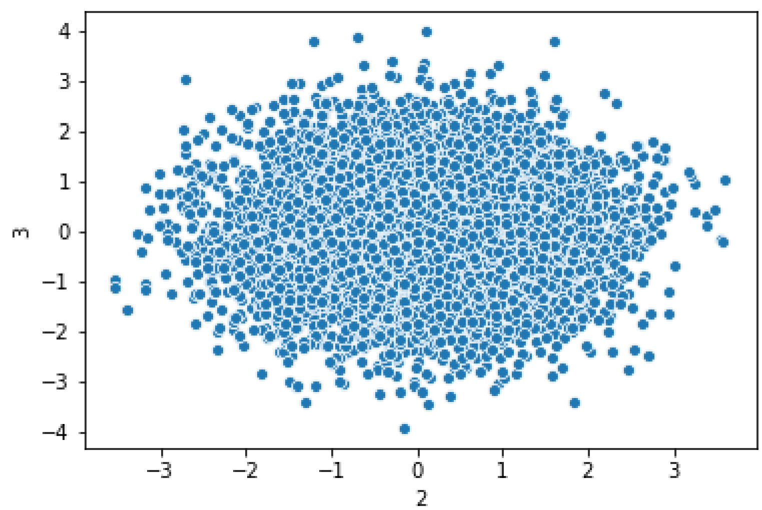
In [11]:
```
sb.scatterplot(data=df,x='1',y='3')
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1efc0828>



In [12]:
```
sb.scatterplot(data=df,x='2',y='3')
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1f0a0780>



(b) Compute the covariance matrix of the data. Write the explicit expression for what this is, and then use any command you like (e.g., np.cov) to compute the 4 × 4 matrix. Explain why the numbers that you get fit with the plots you got. cov(X,Y) = sum[i=1 -> n] ((Xi - X)(Yi - Y)) / (n - 1)

The covariance allows us to see if one variable increases or decreases in accordance to another variable to an extent. The extent of the correlation is measured from -1 to +1, -1 being fully negatively correlated and +1 being fully positively correlated.

In [13]:
```
np.cov(df.T)
```

Out[13]:
```
array([[ 1.00155793, -0.00401176,  0.99162409,  0.00412485],
       [-0.00401176,  1.00537841, -0.00409877, -0.99545662],
       [ 0.99162409, -0.00409877,  1.00158867,  0.00408108],
       [ 0.00412485, -0.99545662,  0.00408108,  1.00516828]])
```

cov(0, 1, 2, 3) =

[var(0), cov(0,1), cov(0,2), cov(0,3)

. cov(1,0), var(1), cov(1,2), cov(1,3)

. cov(2,0), cov(2,1), var(2), cov(2,3)

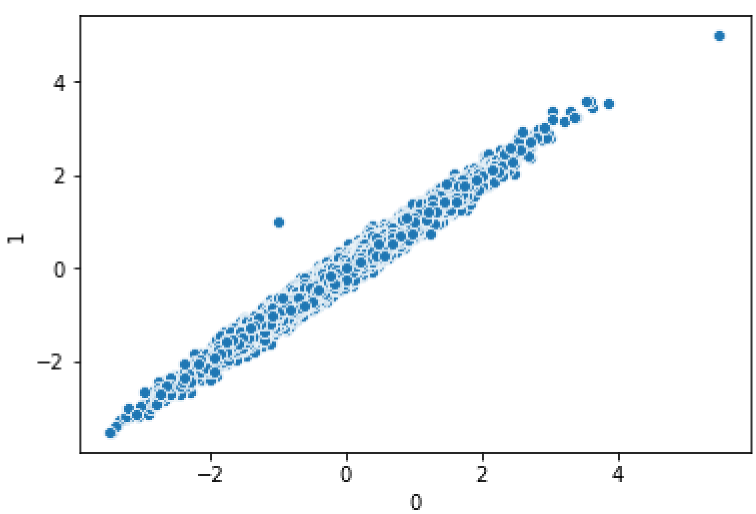
. cov(3,0), cov(3,1), cov(3,2), var(3)]

These numbers correlate with the plots looking at the plots with very strong correlations. In question 1.a, columns 0 & 2 and 1 & 3 seemed very strongly correlated. Looking at the labeled matrix above, cov(1,3) (also equal to cov(3,1)) is very close to -1, meaning that it is almost perfectly, negatively correlated which is shown in the plots. cov(0,2) close to 1, meaning that is it s almost perfectly, positively correlated which is shown in the plots above.

Question 2

In [76]:
```
df2 = pd.read_csv("DF2", header=0, na_values='?', index_col=0)
sb.scatterplot(data=df2,x='0',y='1')
```

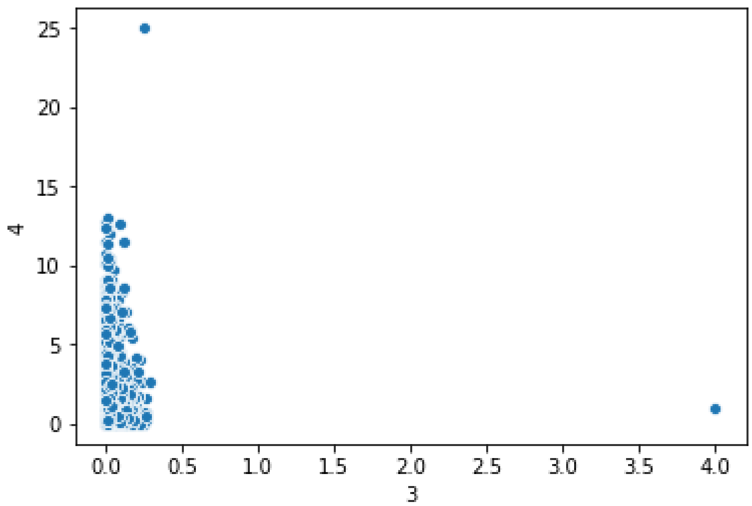
Out[76]: <matplotlib.axes._subplots.AxesSubplot at 0x1a3bcc5160>



The above plot shows that we have 2 outliers -> (-1,1) and (5.5,5) To bring out the fact that the first outlier is actually more farther away from the second one, we actually try to get the square of the difference between the 2 columns of the dataset and then plot them.

In [77]:
```
df3 = df2
df3['3'] = df2['0'] - df2['1']
df3['4'] = df2['1']
df4 = df3.apply(np.square)
sb.scatterplot(data=df4,x='3',y='4')
```

Out[77]: <matplotlib.axes._subplots.AxesSubplot at 0x1a27136cf8>



Question 4

The goal of this exercise is for you to get more experience with Pandas, and to get a chance to explore a cool data set. Download the file Names.zip from Canvas. This contains the frequency of all names that appeared more than 5 times on a social security application from 1880 through 2015. • It could be that names are more diverse now than they were in 1880, so that a name may be relatively the most popular, though its frequency may have been decreasing over the years. Modify the above to return the relative frequency. Note that in the next coming lectures we will learn how to quantify diversity using entropy. • Find all the names that used to be more popular for one gender, but then became more popular for another gender. • (Optional) Find something cool about this data set.

In [33]:
```
import glob, os, csv
cwd = os.getcwd()#os.chdir("/Names")
names_dir = os.path.join(cwd, 'Names', '*.txt')

df = pd.DataFrame({'Name':pd.Series([], dtype='str'),
                   'Gender':pd.Series([], dtype='str'),
                   'Frequency':pd.Series([], dtype='int'),
                   'Year':pd.Series([], dtype='int')})

for file in glob.glob(names_dir):
    year = int(os.path.basename(file).split('.')[0][-4:])
    dataframe_temp = pd.read_csv(file, header=None) #read the csv
    dataframe_temp.columns=['Name', 'Gender','Frequency']
    dataframe_temp['Year']=year
    df = df.append(dataframe_temp,ignore_index=True)

#df
```

Write a program that on input k and XXXX, returns the top k names from year XXXX.

In [35]:
```
def topnames_k_xxxx(k,xxxx,df):
    #xxxx = year, k = how many rows to return
    df_func = df[df.Year == xxxx]
    df_func = df_func.sort_values(by=['Frequency'],ascending=False)
    return df_func.head(k)

topnames_k_xxxx(2,1880,df)
```

Out[35]:

	Name	Gender	Frequency	Year
1774594	John	M	9655	1880
1774595	William	M	9531	1880

Write a program that on input Name returns the frequency for men and women of the name

In [36]:
```
def name_freq_gender(name,df):
    df_male = df[df.Gender == "M"]
    df_male = df_male[df.Name == name]
    name_frequency_male = df_male['Frequency'].sum()

    df_female = df[df.Gender == "F"]
    df_female = df_female[df.Name == name]
    name_frequency_female = df_female['Frequency'].sum()

    return ("Male: "+str(name_frequency_male)+" | "+"Female: "+str(name_
frequency_female))

name_freq_gender('Mary',df)
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: UserWar
ning: Boolean Series key will be reindexed to match DataFrame index.
  This is separate from the ipykernel package so we can avoid doing imp
orts until
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:7: UserWar
ning: Boolean Series key will be reindexed to match DataFrame index.
  import sys
```

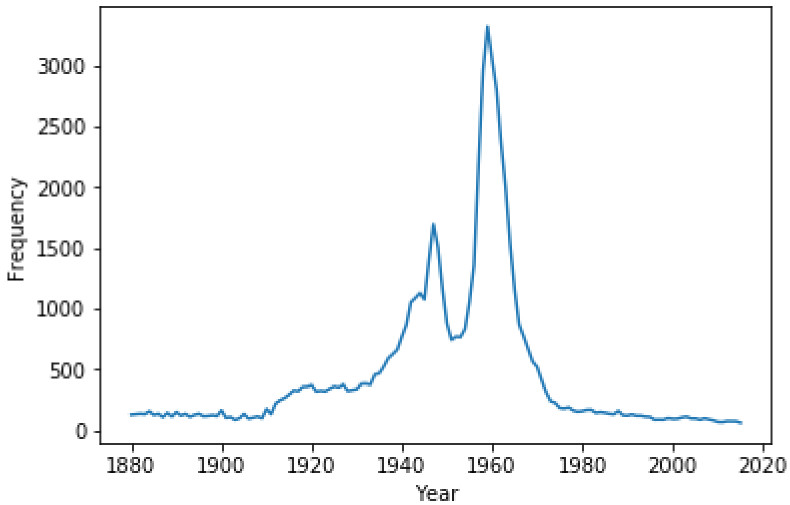
Out[36]:
```
'Male: 15158 | Female: 4118058'
```

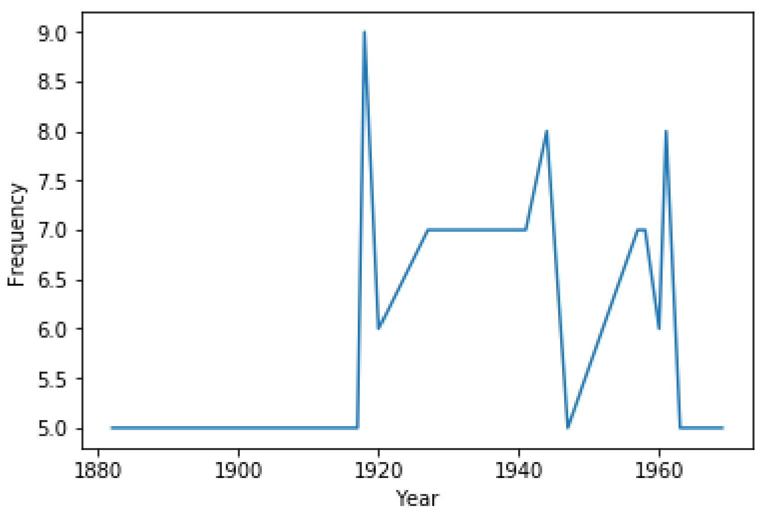
It could be that names are more diverse now than they were in 1880, so that a name may be relatively the most popular, though its frequency may have been decreasing over the years. Modify the above to return the relative frequency. Note that in the next coming lectures we will learn how to quantify diversity using entropy.

In [38]:
```
def name_freq_gender_frequency_timeseries(name,df):
    df_male = df[df.Gender == "M"]
    df_male = df_male[df.Name == name]
    df_male = df_male.sort_values(by=['Year'])
    name_frequency_male = df_male['Frequency'].sum()

    df_female = df[df.Gender == "F"]
    df_female = df_female[df.Name == name]
    df_female = df_female.sort_values(by=['Year'])
    name_frequency_female = df_female['Frequency'].sum()

    fig = plt.figure()
    plt.plot(df_male['Year'], df_male['Frequency'])
    plt.xlabel("Year")
    plt.ylabel("Frequency")

    fig2 = plt.figure()
    plt.plot(df_female['Year'], df_female['Frequency'])
    plt.xlabel("Year")
    plt.ylabel("Frequency")

    return plt


name_freq_gender_frequency_timeseries('Dave', df)
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: UserWar
ning: Boolean Series key will be reindexed to match DataFrame index.
  This is separate from the ipykernel package so we can avoid doing imp
orts until
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:8: UserWar
ning: Boolean Series key will be reindexed to match DataFrame index.
```

Out[38]: `<module 'matplotlib.pyplot' from '/anaconda3/lib/python3.6/site-packages/matplotlib/pyplot.py'>`





In [39]:
```
def name_shift(df):
    a = df.copy()
    a['1/2'] = np.where(a['Year'] <= 1947, '1','2')
    b = a.groupby(['Name', '1/2', 'Gender']).sum().reset_index()
    b = b.groupby('Name').filter(lambda x: len(x) == 4)
    cp = b[b['1/2'] == '1'][['Name', 'Gender', 'Frequency']]
    cpm = cp[cp['Gender'] == 'M']
    cpf = cp[cp['Gender'] == 'F']

    cn = b[b['1/2'] == '2'][['Name', 'Gender', 'Frequency']]
    cnm = cn[cn['Gender'] == 'M']
    cnf = cn[cn['Gender'] == 'F']

    males = pd.merge(cpm, cnm, on=['Name', 'Gender'])
    females = pd.merge(cpf, cnf, on=['Name', 'Gender'])

    males['increasing'] = np.where(males['Frequency_x'] < males['Frequen
cy_y'], True, False)
    females['increasing'] = np.where(females['Frequency_x'] < females['F
requency_y'], True, False)
    return males[males['increasing'] != females['increasing']]['Name'].t
olist()



name_shift(df)
```

```
Out[39]: ['Abbie',
          'Abie',
          'Adie',
          'Afton',
          'Al',
          'Alba',
          'Alder',
          'Allie',
          'Allison',
          'Alpha',
          'Amie',
          'Amparo',
          'Ange',
          'Angelita',
          'Ann',
          'Aquilla',
          'Ara',
          'Arden',
          'Arizona',
          'Arleigh',
          'Arley',
          'Arlin',
          'Arlyn',
          'Arlynn',
          'Arnell',
          'Arnett',
          'Arnie',
          'Artie',
          'Artis',
          'Aster',
          'Asuncion',
          'Atlas',
          'Aubra',
          'Aubrey',
          'Aubry',
          'Auburn',
          'Audie',
          'Audra',
          'Audrey',
          'Audry',
          'August',
          'Auguste',
          'Augustine',
          'Aurora',
          'Averil',
          'Averill',
          'Avie',
          'Avis',
          'Basil',
          'Bee',
          'Bennette',
          'Benny',
          'Berlin',
          'Berlyn',
          'Bernell',
          'Bernie',
          'Berry',
```

```
'Betsy',
'Billy',
'Bliss',
'Bobbie',
'Bobby',
'Bonnie',
'Bonny',
'Buddy',
'Bunny',
'Callie',
'Camille',
'Cammie',
'Carle',
'Carlee',
'Carley',
'Carlie',
'Carlisle',
'Carlyle',
'Carlyn',
'Carman',
'Carmen',
'Carmin',
'Carmon',
'Carnell',
'Carol',
'Carrie',
'Carry',
'Cassie',
'Challis',
'Charl',
'Charles',
'Charley',
'Charlie',
'Charlotte',
'Cherry',
'Chesley',
'Christie',
'Clair',
'Claire',
'Clare',
'Clary',
'Claudell',
'Claudette',
'Clay',
'Cleaster',
'Cleotha',
'Clifton',
'Clover',
'Clydell',
'Colie',
'Conda',
'Connie',
'Constance',
'Consuelo',
'Coral',
'Cordell',
'Corliss',
```

```
'Cornelius',
'Cornell',
'Cortez',
'Coy',
'Cuba',
'Dabney',
'Dail',
'Dannie',
'Dare',
'Daris',
'Darris',
'Daun',
'Dave',
'Davie',
'Day',
'Dea',
'Dean',
'Deanie',
'Dee',
'Delaine',
'Delano',
'Delia',
'Dell',
'Delone',
'Delta',
'Dempsey',
'Dennie',
'Denzel',
'Derryl',
'Donie',
'Donnie',
'Donnis',
'Donzell',
'Dorie',
'Douglass',
'Dove',
'Duane',
'Eddie',
'Edie',
'Eileen',
'Elba',
'Eliza',
'Elize',
'Ella',
'Ellen',
'Ellie',
'Ellis',
'Elver',
'Elvia',
'Elvis',
'Emer',
'Emerald',
'Emile',
'Emma',
'Emmett',
'Emory',
'Eppie',
```

```
'Erie',
'Ernie',
'Esperanza',
'Estee',
'Eugenia',
'Everett',
'Everette',
'Evern',
'Evie',
'Faris',
'Fayette',
'Felice',
'Ferris',
'Fletcher',
'Forrest',
'Foster',
'Frankie',
'Freddie',
'Freddy',
'Gail',
'Gale',
'Garnell',
'Gay',
'Gayle',
'Gaynor',
'Genie',
'Georg',
'Gerry',
'Glen',
'Glendell',
'Glenn',
'Glynn',
'Grace',
'Grady',
'Guy',
'Gwen',
'Gwin',
'Gwyn',
'Gwynn',
'Halley',
'Hallie',
'Happy',
'Harlan',
'Harlie',
'Harmon',
'Harris',
'Henri',
'Henrie',
'Hilary',
'Hillary',
'Holley',
'Hollie',
'Hollis',
'Hope',
'Hurley',
'Idris',
'Iris',
```

```
 'Isabel',
 'Ivery',
 'Ivey',
 'Ivie',
 'Ivy',
 'Jakie',
 'Jean',
 'Jeanne',
 'Jene',
 'Jeral',
 'Jerrel',
 'Jerris',
 'Jerry',
 'Jessie',
 'Jim',
 'Joan',
 'Joanne',
 'Joda',
 'Johan',
 'Johann',
 'John',
 'Josefina',
 'Joy',
 'Juel',
 'Julius',
 'Kaye',
 'Kelsie',
 'King',
 'Lacy',
 'Ladell',
 'Lafayette',
 'Lanie',
 'Lanis',
 'Lannie',
 'Lannis',
 'Lark',
 'Laurel',
 'Lauri',
 'Laurice',
 'Laurie',
 'Laurin',
 'Lavaughn',
 'Lavell',
 'Lavelle',
 'Lavon',
 'Lavone',
 'Lavonne',
 'Lea',
 'Ledell',
 'Lemon',
 'Lennell',
 'Lennie',
 'Lennis',
 'Leone',
 'Leonor',
 'Leslie',
 'Levern',
```

```
'Lew',
'Lexie',
'Lind',
'Lindley',
'Lindy',
'Lisle',
'Lonnie',
'Lora',
'Lory',
'Lou',
'Love',
'Lovell',
'Loyal',
'Lu',
'Lugene',
'Lyndall',
'Lyndell',
'Lynne',
'Mabry',
'Mac',
'Macey',
'Mackie',
'Maitland',
'Major',
'Marcell',
'Marcelle',
'March',
'Mardell',
'Marlene',
'Marlyn',
'Marne',
'Marvel',
'Marvell',
'Marvis',
'Maryann',
'Marzell',
'Matilde',
'Matty',
'Maurice',
'Merced',
'Meredith',
'Meridith',
'Merlin',
'Merril',
'Merrill',
'Merritt',
'Meryl',
'Mikie',
'Miley',
'Mona',
'Monnie',
'Monroe',
'Monta',
'Montie',
'Morley',
'Myron',
'Nadine',
```

```
 'Naomi',
 'Neal',
 'Nealy',
 'Nebraska',
 'Neddie',
 'Neely',
 'Nick',
 'Nora',
 'Norah',
 'Noris',
 'Oliver',
 'Omega',
 'Oren',
 'Orla',
 'Ozzie',
 'Paddy',
 'Palmer',
 'Parnell',
 'Patrica',
 'Patty',
 'Paulette',
 'Peggy',
 'Pernell',
 'Petra',
 'Phil',
 'Posey',
 'Prentice',
 'Prince',
 'Rae',
 'Rainey',
 'Raleigh',
 'Raphael',
 'Rea',
 'Redell',
 'Refugio',
 'Rena',
 'Reo',
 'Reuben',
 'Reyes',
 'Rhea',
 'Richie',
 'Ritchie',
 'Robbie',
 'Roby',
 'Roe',
 'Romaine',
 'Romell',
 'Romie',
 'Rosa',
 'Rosario',
 'Rosemarie',
 'Roslyn',
 'Roxy',
 'Royal',
 'Rozell',
 'Rozelle',
 'Rudie',
```

```
'Rue',
'Sadie',
'Sally',
'Salvatore',
'Samie',
'Santos',
'Selby',
'Selwyn',
'Shellie',
'Sherald',
'Sherill',
'Sherley',
'Sherman',
'Sherrel',
'Sherrell',
'Sherril',
'Sherrill',
'Shirl',
'Sidney',
'Sophie',
'Starley',
'Starling',
'Sunnie',
'Sydney',
'Tallie',
'Ted',
'Teddie',
'Teddy',
'Temple',
'Texas',
'Tharon',
'Theo',
'Theodore',
'Tobe',
'Toy',
'Trinidad',
'Valentine',
'Vance',
'Vennie',
'Ventura',
'Venus',
'Verdell',
'Vernell',
'Vinnie',
'Vondell',
'Wallis',
'Wally',
'Ward',
'Warnell',
'Waverly',
'Wendolyn',
'Wiley',
'Willia',
'William',
'Willy',
'Winston',
'Woody',
```

```
 'Wray',
 'Ysabel',
 'Zenith',
 'Zina']
```

Question 5

In [1]:
```
import pandas as pd
tweets = pd.read_csv("tweets.csv")
tweets.head()
```

Out[1]:

	id	id_str	user_location	user_bg_color	retweet_count	user_name	
0	1	729828033092149248	Wheeling WV	022330	0	Jaybo26003	(
1	2	729828033092161537	NaN	C0DEED	0	britttttany_ns	(
2	3	729828033566224384	NaN	C0DEED	0	JeffriesLori	(
3	4	729828033893302272	global	C0DEED	0	WhorunsGOVs	(
4	5	729828034178482177	California, USA	131516	0	BJCG0830	(

In [2]:
```
tweets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237480 entries, 0 to 237479
Data columns (total 15 columns):
id                    237480 non-null int64
id_str                237480 non-null int64
user_location         153514 non-null object
user_bg_color         237480 non-null object
retweet_count         237480 non-null int64
user_name             237480 non-null object
polarity              237480 non-null float64
created               237480 non-null object
geo                   125 non-null object
user_description      191850 non-null object
user_created          237480 non-null object
user_followers        237480 non-null int64
coordinates           125 non-null object
subjectivity          237480 non-null float64
text                  237480 non-null object
dtypes: float64(2), int64(4), object(9)
memory usage: 27.2+ MB
```

In [3]:
```
def get_candidates(row):
    candidates = []
    text = row["text"].lower()
    if "clinton" in text or "hillary" in text:
        candidates.append("clinton")
    if "trump" in text or "donald" in text:
        candidates.append("trump")
    if "sanders" in text or "bernie" in text:
        candidates.append("sanders")
    return ",".join(candidates)

tweets["candidates"] = tweets.apply(get_candidates, axis = 1)
```

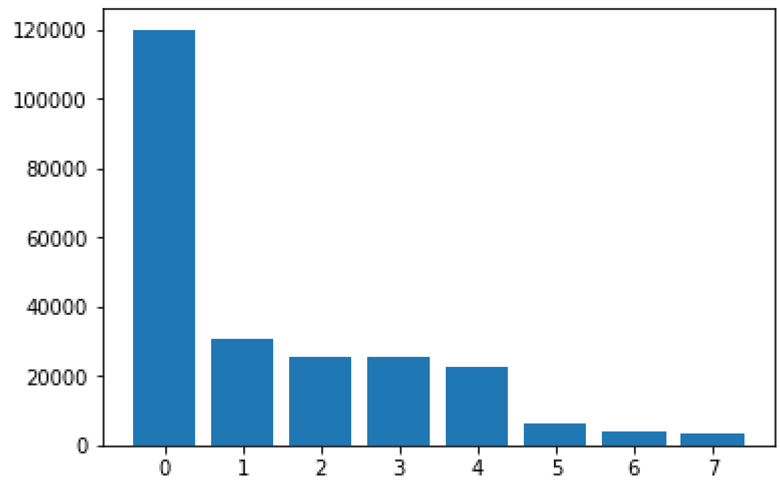
In [4]:
```
tweets.head()
```

Out[4]:

	id	id_str	user_location	user_bg_color	retweet_count	user_name	(
0	1	729828033092149248	Wheeling WV	022330	0	Jaybo26003	(
1	2	729828033092161537	NaN	C0DEED	0	britttttany_ns	(
2	3	729828033566224384	NaN	C0DEED	0	JeffriesLori	(
3	4	729828033893302272	global	C0DEED	0	WhorunsGOVs	(
4	5	729828034178482177	California, USA	131516	0	BJCG0830	(

In [5]:
```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

In [6]:
```
counts = tweets["candidates"].value_counts()
plt.bar(range(len(counts)), counts)
plt.show()

print(counts)
```



```
trump                      119998
clinton,trump               30521
                            25429
sanders                     25351
clinton                     22746
clinton,sanders              6044
clinton,trump,sanders        4219
trump,sanders                3172
Name: candidates, dtype: int64
```

In [7]:
```
from datetime import datetime
```

In [8]:
```
tweets["created"]
```

```
Out[8]: 0         2016-05-10T00:18:57
        1         2016-05-10T00:18:57
        2         2016-05-10T00:18:57
        3         2016-05-10T00:18:57
        4         2016-05-10T00:18:57
        5         2016-05-10T00:18:52
        6         2016-05-10T00:18:57
        7         2016-05-10T00:18:57
        8         2016-05-10T00:18:57
        9         2016-05-10T00:18:57
        10        2016-05-10T00:18:57
        11        2016-05-10T00:18:57
        12        2016-05-10T00:18:57
        13        2016-05-10T00:18:57
        14        2016-05-10T00:18:57
        15        2016-05-10T00:18:57
        16        2016-05-10T00:18:57
        17        2016-05-10T00:18:57
        18        2016-05-10T00:18:57
        19        2016-05-10T00:18:57
        20        2016-05-10T00:18:57
        21        2016-05-10T00:18:58
        22        2016-05-10T00:18:58
        23        2016-05-10T00:18:58
        24        2016-05-10T00:18:58
        25        2016-05-10T00:18:58
        26        2016-05-10T00:18:58
        27        2016-05-10T00:18:58
        28        2016-05-10T00:18:58
        29        2016-05-10T00:18:58
                          ...
        237450    2016-05-10T17:44:31
        237451    2016-05-10T17:44:31
        237452    2016-05-10T17:44:31
        237453    2016-05-10T17:44:31
        237454    2016-05-10T17:44:31
        237455    2016-05-10T17:44:31
        237456    2016-05-10T17:44:31
        237457    2016-05-10T17:44:32
        237458    2016-05-10T17:44:31
        237459    2016-05-10T17:44:31
        237460    2016-05-10T17:44:32
        237461    2016-05-10T17:44:32
        237462    2016-05-10T17:44:32
        237463    2016-05-10T17:44:32
        237464    2016-05-10T17:44:32
        237465    2016-05-10T17:44:32
        237466    2016-05-10T17:44:32
        237467    2016-05-10T17:44:32
        237468    2016-05-10T17:44:32
        237469    2016-05-10T17:44:32
        237470    2016-05-10T17:44:32
        237471    2016-05-10T17:44:32
        237472    2016-05-10T17:44:32
        237473    2016-05-10T17:44:32
        237474    2016-05-10T17:44:32
        237475    2016-05-10T17:44:32
```

```
237476    2016-05-10T17:44:32
237477    2016-05-10T17:44:32
237478    2016-05-10T17:44:32
237479    2016-05-10T17:44:32
Name: created, Length: 237480, dtype: object
```

In [9]:
```
tweets["created"] = pd.to_datetime(tweets["created"])
```

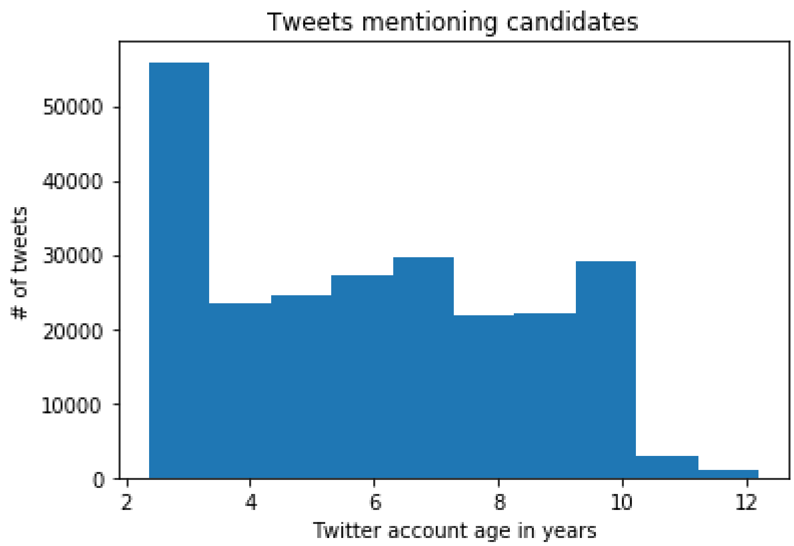
In [10]:
```
tweets["user_created"] = pd.to_datetime(tweets["user_created"])
```

In [11]:
```
tweets["user_age"] = tweets["user_created"].apply(lambda x: (datetime.no
w() - x).total_seconds() / 3600 / 24 / 365)
tweets["user_age"]
```

```
Out[11]: 0          6.847500
         1          5.741702
         2          5.944790
         3          4.594896
         4          9.507893
         5          5.882978
         6          3.072639
         7          3.212347
         8          8.801344
         9          5.620845
         10         2.505061
         11         9.456094
         12         8.396189
         13         4.562284
         14         5.933056
         15         3.760194
         16         8.154277
         17         2.969259
         18         3.217352
         19         2.642914
         20         2.807074
         21         2.432229
         22         5.017705
         23         2.570974
         24         7.325041
         25         2.602256
         26         2.950162
         27         8.635029
         28         8.204997
         29         2.694239
                      ...
         237450     9.225565
         237451     2.528622
         237452     6.662476
         237453     6.979823
         237454     6.883152
         237455     5.836273
         237456     4.486457
         237457     7.410809
         237458     6.539180
         237459     9.402098
         237460     2.458281
         237461     6.500517
         237462     3.446337
         237463     6.963247
         237464     4.963705
         237465     6.438565
         237466     2.486724
         237467     7.452479
         237468    10.094304
         237469     5.271025
         237470     2.826798
         237471    10.514237
         237472     9.429395
         237473     9.009381
         237474     3.122193
         237475     4.753370
```

```
237476     3.026865
237477     7.455798
237478     6.674787
237479     8.861932
Name: user_age, Length: 237480, dtype: float64
```

In [12]:
```
plt.hist(tweets["user_age"])
plt.title("Tweets mentioning candidates")
plt.xlabel("Twitter account age in years")
plt.ylabel("# of tweets")
plt.show()
```



In [13]:
```
tweets.head()
```

Out[13]:

	id	id_str	user_location	user_bg_color	retweet_count	user_name	(
0	1	729828033092149248	Wheeling WV	022330	0	Jaybo26003	(
1	2	729828033092161537	NaN	C0DEED	0	britttttany_ns	(
2	3	729828033566224384	NaN	C0DEED	0	JeffriesLori	(
3	4	729828033893302272	global	C0DEED	0	WhorunsGOVs	(
4	5	729828034178482177	California, USA	131516	0	BJCG0830	(

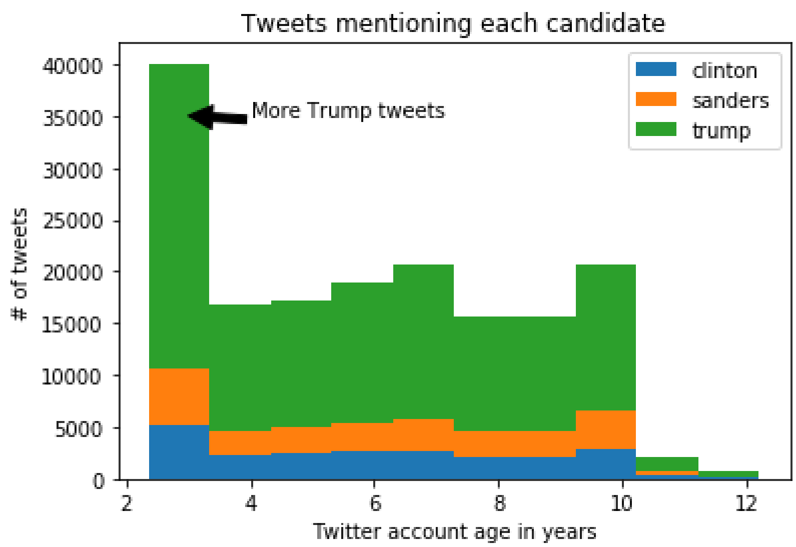
In [14]:
```
cl_tweets = tweets["user_age"][tweets["candidates"] == "clinton"]
sa_tweets = tweets["user_age"][tweets["candidates"] == "sanders"]
tr_tweets = tweets["user_age"][tweets["candidates"] == "trump"]
```

In [15]:
```
cl_tweets
```

```
Out[15]: 5           5.882978
         9           5.620845
         23          2.570974
         32          2.422857
         34          8.302552
         40          8.776823
         51          2.686604
         57          3.294067
         61          5.319723
         78          7.820821
         83          7.189833
         100         7.158588
         127         2.385138
         140         9.048246
         149         3.099776
         154         6.577110
         156         2.549581
         159         2.620740
         166         8.210267
         181         9.392824
         189         9.191029
         198         7.648333
         212         9.252838
         213         7.268262
         227         7.308706
         228         8.456196
         229         9.560341
         235         6.847158
         236         8.056988
         240         7.154157
                       ...
         237164      3.152417
         237167      4.787311
         237175      5.092435
         237183      2.369336
         237186      9.425469
         237188      2.664243
         237206      7.125034
         237212      5.795405
         237218      9.236498
         237240      5.593830
         237242      4.366406
         237245      3.699843
         237290      2.800678
         237302      4.816216
         237303      5.469419
         237304      9.304744
         237309      6.040117
         237314      9.420955
         237322      2.844952
         237327      3.336995
         237330      5.889283
         237340      2.578941
         237342      2.391569
         237349     10.519524
         237381      3.492605
         237400      7.253667
```

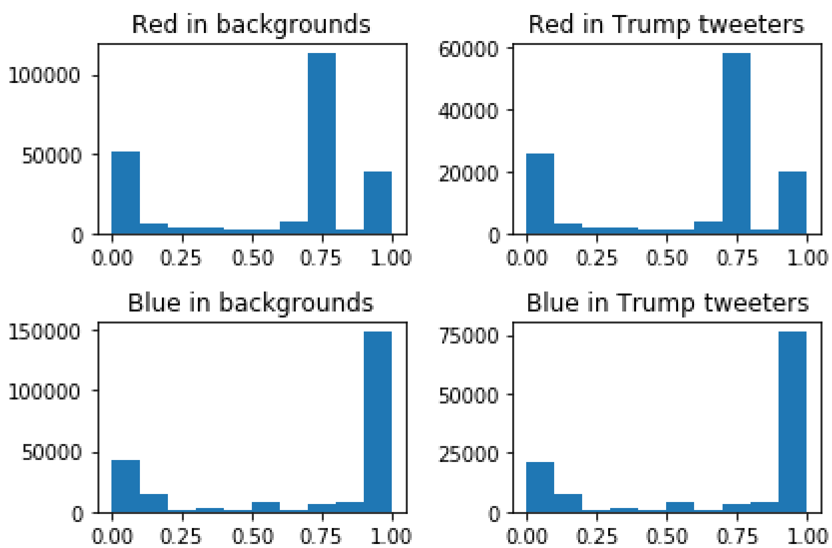
```
237415     2.616143
237439     2.488139
237463     6.963247
237471    10.514237
Name: user_age, Length: 22746, dtype: float64
```

In [16]:
```
plt.hist([
        cl_tweets,
        sa_tweets,
        tr_tweets
    ],
    stacked=True,
    label=["clinton", "sanders", "trump"]
)
plt.legend()
plt.title("Tweets mentioning each candidate")
plt.xlabel("Twitter account age in years")
plt.ylabel("# of tweets")
plt.annotate('More Trump tweets', xy=(3, 35000), xytext=(4, 35000),
            arrowprops=dict(facecolor='black'))
plt.show()
```



In [17]:
```
import matplotlib.colors as colors

tweets["red"] = tweets["user_bg_color"].apply(lambda x: colors.hex2color
('#{0}'.format(x))[0])
tweets["blue"] = tweets["user_bg_color"].apply(lambda x: colors.hex2colo
r('#{0}'.format(x))[2])
```

In [18]:
```
fig, axes = plt.subplots(nrows=2, ncols=2)
ax0, ax1, ax2, ax3 = axes.flat

ax0.hist(tweets["red"])
ax0.set_title('Red in backgrounds')

ax1.hist(tweets["red"][tweets["candidates"] == "trump"].values)
ax1.set_title('Red in Trump tweeters')

ax2.hist(tweets["blue"])
ax2.set_title('Blue in backgrounds')

ax3.hist(tweets["blue"][tweets["candidates"] == "trump"].values)
ax3.set_title('Blue in Trump tweeters')

plt.tight_layout()
plt.show()
```

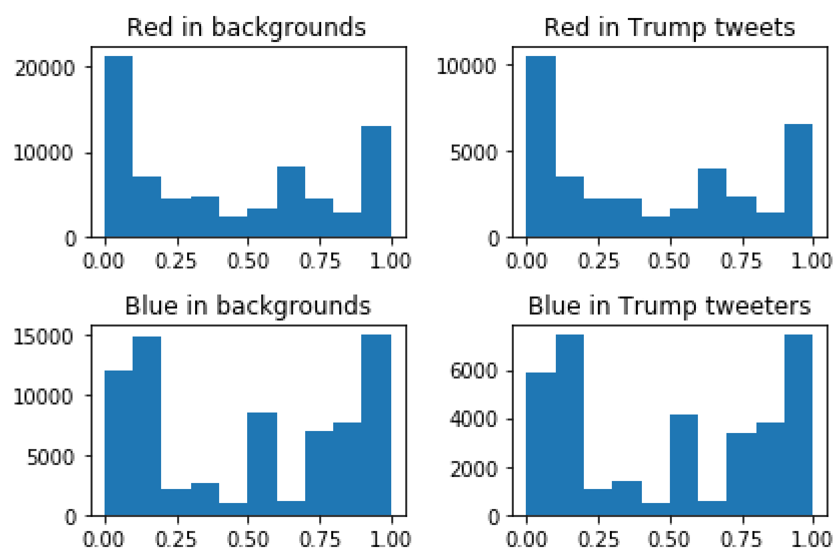


In [19]:
```
tweets["user_bg_color"].value_counts()
```

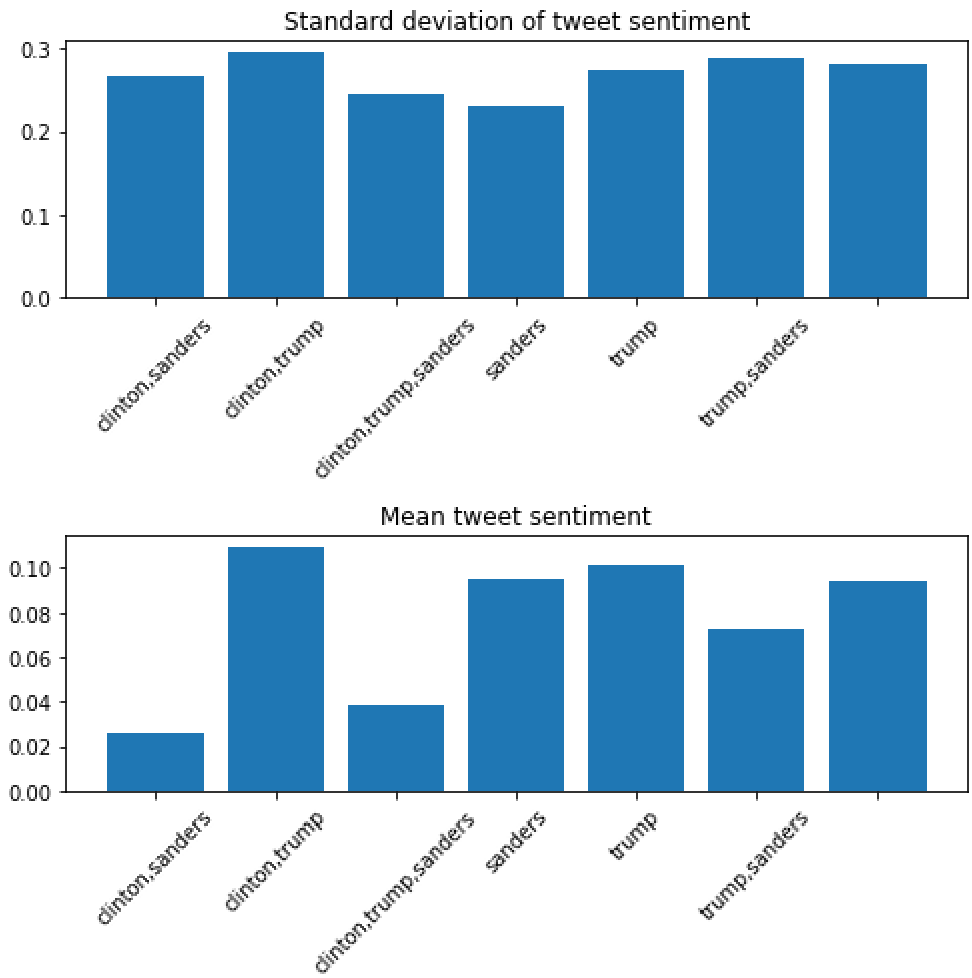
```
Out[19]: C0DEED    108977
         000000     31119
         F5F8FA     25597
         131516      7731
         1A1B1F      5059
         022330      4300
         0099B9      3958
         642D8B      3767
         FFFFFF      3101
         9AE4E8      2651
         ACDED6      2383
         352726      2338
         C6E2EE      1978
         709397      1518
         EBEBEB      1475
         FF6699      1370
         BADFCD      1336
         FFF04D      1300
         EDECE9      1225
         B2DFDA      1218
         DBE9ED      1113
         ABB8C2      1101
         8B542B      1073
         3B94D9       623
         89C9FA       414
         DD2E44       351
         94D487       318
         4A913C       300
         9266CC       287
         F5ABB5       267
                    ...
         E31212         1
         26C926         1
         00D277         1
         E0E094         1
         335577         1
         85FFE7         1
         A8455E         1
         061114         1
         DE16BD         1
         96B3AE         1
         1316E5         1
         837D75         1
         00FFEA         1
         8CBAA3         1
         FF0F0F         1
         275579         1
         FF9305         1
         01C57B         1
         FFE400         1
         F5E7D2         1
         07022E         1
         00B88A         1
         010308         1
         A13D18         1
         096BE3         1
         801010         1
```

```
1B4873          1
9F00B8          1
33D413          1
EDF508          1
Name: user_bg_color, Length: 6970, dtype: int64
```

In [20]:
```
tc = tweets[~tweets["user_bg_color"].isin(["C0DEED", "000000", "F5F8FA"
])]

def create_plot(data):
    fig, axes = plt.subplots(nrows=2, ncols=2)
    ax0, ax1, ax2, ax3 = axes.flat

    ax0.hist(data["red"])
    ax0.set_title('Red in backgrounds')

    ax1.hist(data["red"][data["candidates"] == "trump"].values)
    ax1.set_title('Red in Trump tweets')

    ax2.hist(data["blue"])
    ax2.set_title('Blue in backgrounds')

    ax3.hist(data["blue"][data["candidates"] == "trump"].values)
    ax3.set_title('Blue in Trump tweeters')

    plt.tight_layout()
    plt.show()

create_plot(tc)
```

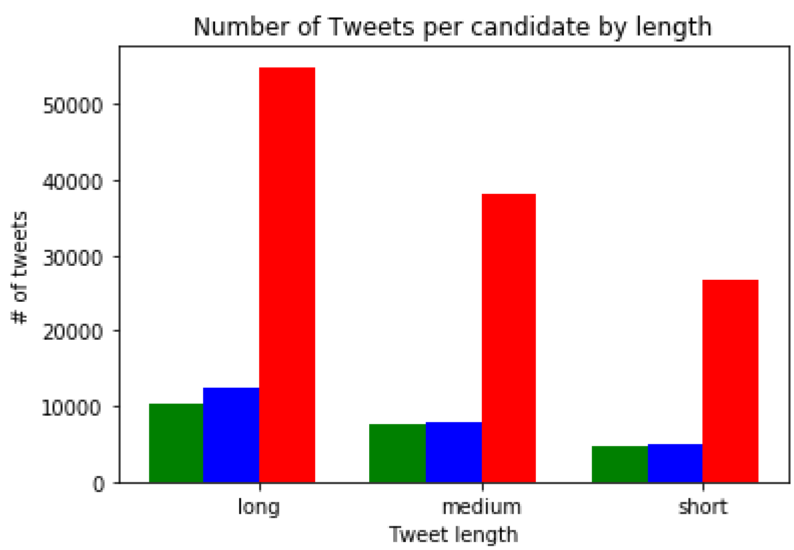


In [21]:
```
gr = tweets.groupby("candidates").agg([np.mean, np.std])

fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(7, 7))
ax0, ax1 = axes.flat

std = gr["polarity"]["std"].iloc[1:]
mean = gr["polarity"]["mean"].iloc[1:]
ax0.bar(range(len(std)), std)
ax0.set_xticklabels(std.index, rotation=45)
ax0.set_title('Standard deviation of tweet sentiment')

ax1.bar(range(len(mean)), mean)
ax1.set_xticklabels(mean.index, rotation=45)
ax1.set_title('Mean tweet sentiment')

plt.tight_layout()
plt.show()
```

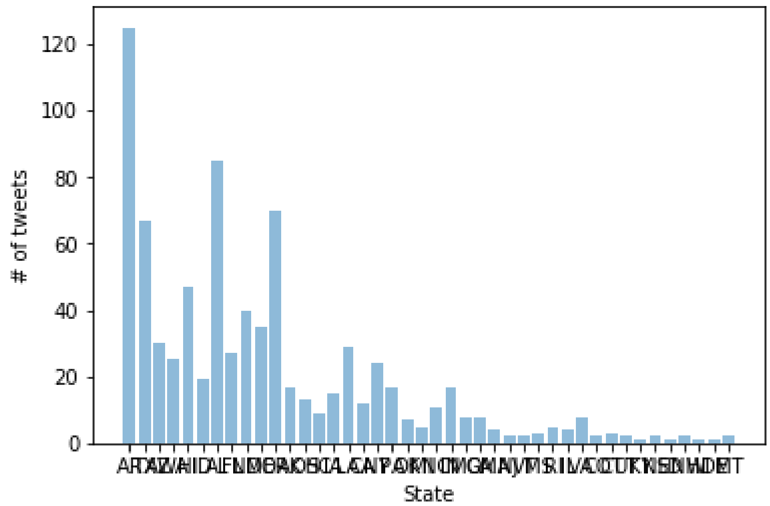


In [22]:
```
def tweet_lengths(text):
    if len(text) < 100:
        return "short"
    elif 100 <= len(text) <= 135:
        return "medium"
    else:
        return "long"

tweets["tweet_length"] = tweets["text"].apply(tweet_lengths)

tl = {}
for candidate in ["clinton", "sanders", "trump"]:
    tl[candidate] = tweets["tweet_length"][tweets["candidates"] == candi
date].value_counts()
```

In [23]:
```
fig, ax = plt.subplots()
width = .5
x = np.array(range(0, 6, 2))
ax.bar(x, tl["clinton"], width, color='g')
ax.bar(x + width, tl["sanders"], width, color='b')
ax.bar(x + (width * 2), tl["trump"], width, color='r')

ax.set_ylabel('# of tweets')
ax.set_title('Number of Tweets per candidate by length')
ax.set_xticks(x + (width * 1.5))
ax.set_xticklabels(('long', 'medium', 'short'))
ax.set_xlabel('Tweet length')
plt.show()
```



In [24]:
```
state_abbr = ["AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "FL", "GA"
,
          "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME", "MD",
          "MA", "MI", "MN", "MS", "MO", "MT", "NE", "NV", "NH", "NJ",
          "NM", "NY", "NC", "ND", "OH", "OK", "OR", "PA", "RI", "SC",
          "SD", "TN", "TX", "UT", "VT", "VA", "WA", "WV", "WI", "WY"]

state_names = ["Alabama","Alaska","Arizona","Arkansas","California","Col
orado",
  "Connecticut","Delaware","Florida","Georgia","Hawaii","Idaho","Illinoi
s",
  "Indiana","Iowa","Kansas","Kentucky","Louisiana","Maine","Maryland",
  "Massachusetts","Michigan","Minnesota","Mississippi","Missouri","Monta
na",
  "Nebraska","Nevada","New Hampshire","New Jersey","New Mexico","New Yor
k",
  "North Carolina","North Dakota","Ohio","Oklahoma","Oregon","Pennsylvan
ia",
  "Rhode Island","South Carolina","South Dakota","Tennessee","Texas","Ut
ah",
  "Vermont","Virginia","Washington","West Virginia","Wisconsin","Wyomin
g"]


name_to_abbr = {state_names[i]: state_abbr[i] for i in range(len(state_n
ames))}
all_states = state_abbr + state_names


import re

t_locs = tweets['user_location'].value_counts()
reg = '.({0}).'.format('|'.join(all_states))
tweets_by_state = {}

for i in range(len(t_locs)):
    srem = re.match(reg,t_locs.index[i])
    if srem:
        state = srem.groups()[0]
        if state in state_names:
            state = name_to_abbr[state]
        if state in tweets_by_state:
            tweets_by_state[state] += t_locs[i]
        else:
            tweets_by_state[state] = t_locs[i]

objs = list(tweets_by_state.keys())
y_pos = np.arange(len(objs))
num_tweets = list(tweets_by_state.values())

plt.bar(y_pos, num_tweets, align='center', alpha=0.5)
plt.xticks(y_pos, objs)

plt.xlabel("State")
plt.ylabel("# of tweets")
plt.show()
```

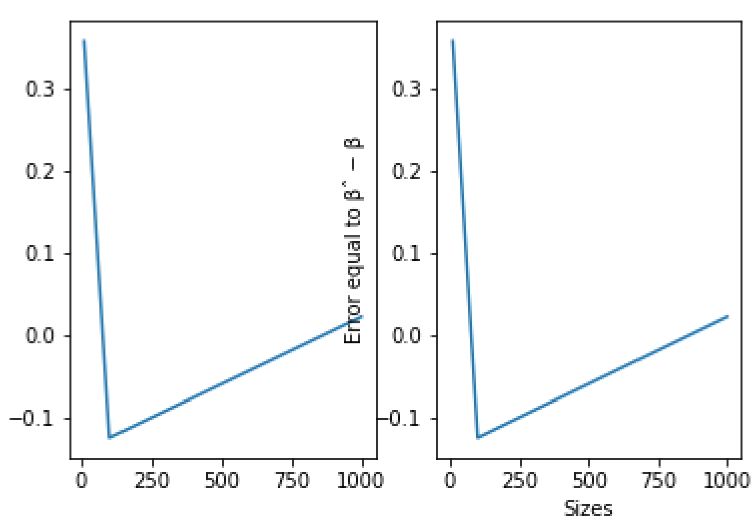


Question 3

In [25]:
```
def CalculatedError(SampleSize):

    N = SampleSize
    Na= (N,1)
    XSmaple = np.random.normal(0, 1, Na)
    ESample = np.random.normal(0,1, Na)

    y =  [(-3 + ESample[i]) for i in range(N)]

    XTrans = np.transpose(XSmaple)

    SampleInv = np.linalg.inv(np.matmul(XTrans,XSmaple ))
    mult_x = np.matmul(XTrans, y)

    Tr_Y = np.matmul(SampleInv,mult_x)
    CalculatedZ = np.matmul(SampleInv,XTrans)

    error = np.matmul(CalculatedZ, ESample)
    print ("The error is :", error)

    return error
```

In [26]:
```
import math
sizes = [int(math.pow(10, i)) for i in range(1, 4)]

error_values = []
for size in sizes:
    print (size)
    error_fun = CalculatedError(size)
    print(error_fun)
    error_values.append(error_fun)

type(error_values)
listobject = [error_values[i].tolist() for i in range(len(error_values
))]
new_list = []
[new_list.append(listobject[i][0]) for i in range(len(error_values))]
print("Values are ", new_list)
fig = plt.figure()

plt.subplot(1, 2, 1)
plt.plot(sizes, new_list)
plt.subplot(1, 2, 2)
plt.plot(sizes, new_list)
for i in sizes:
    plt.plot(pow(i,-0.5))
plt.xlabel("Sizes")
plt.ylabel("Error equal to β^ - β")
plt.show()
```

```
10
The error is : [[0.35691705]]
[[0.35691705]]
100
The error is : [[-0.12512468]]
[[-0.12512468]]
1000
The error is : [[0.02152895]]
[[0.02152895]]
Values are  [[0.35691705041359734], [-0.12512468059479628], [0.02152895
156863943]]
```



Question 6

In [28]:
```
# Read in the airports data.
airports = pd.read_csv("airports.csv", header=None, dtype=str, encoding=
"latin-1")
airports.columns = ["id", "name", "city", "country", "code", "icao", "la
titude", "longitude", "altitude", "offset", "dst", "timezone"]
# Read in the airlines data.
airlines = pd.read_csv("airlines.csv", header=None, dtype=str, encoding=
"latin-1")
airlines.columns = ["id", "name", "alias", "iata", "icao", "callsign",
"country", "active"]
# Read in the routes data.
routes = pd.read_csv("routes.csv", header=None, dtype=str, encoding="lat
in-1")
routes.columns = ["airline", "airline_id", "source", "source_id", "dest"
, "dest_id", "codeshare", "stops", "equipment"]
airports.head()
```

Out[28]:

	id	name	city	country	code	icao	latitude	longitude	altitude	of
0	1	Goroka Airport	Goroka	Papua New Guinea	GKA	AYGA	-6.081689835	145.3919983	5282	1(
1	2	Madang Airport	Madang	Papua New Guinea	MAG	AYMD	-5.207079887	145.7890015	20	1(
2	3	Mount Hagen Kagamuga Airport	Mount Hagen	Papua New Guinea	HGU	AYMH	-5.826789856	144.2960052	5388	1(
3	4	Nadzab Airport	Nadzab	Papua New Guinea	LAE	AYNZ	-6.569803	146.725977	239	1(
4	5	Port Moresby Jacksons International Airport	Port Moresby	Papua New Guinea	POM	AYPY	-9.443380356	147.2200012	146	1(

In [29]:
```
airlines.head()
```

Out[29]:

	id	name	alias	iata	icao	callsign	country	active
0	-1	Unknown	\N	-	NaN	\N	\N	Y
1	1	Private flight	\N	-	NaN	NaN	NaN	Y
2	2	135 Airways	\N	NaN	GNL	GENERAL	United States	N
3	3	1Time Airline	\N	1T	RNX	NEXTIME	South Africa	Y
4	4	2 Sqn No 1 Elementary Flying Training School	\N	NaN	WYT	NaN	United Kingdom	N

In [30]:
```
routes.head()
```

Out[30]:

	airline	airline_id	source	source_id	dest	dest_id	codeshare	stops	equipment
0	2B	410	AER	2965	KZN	2990	NaN	0	CR2
1	2B	410	ASF	2966	KZN	2990	NaN	0	CR2
2	2B	410	ASF	2966	MRV	2962	NaN	0	CR2
3	2B	410	CEK	2968	KZN	2990	NaN	0	CR2
4	2B	410	CEK	2968	OVB	4078	NaN	0	CR2

In [31]:
```
routes = routes[routes["airline_id"] != "\\N"] #ensure only numeric data
```

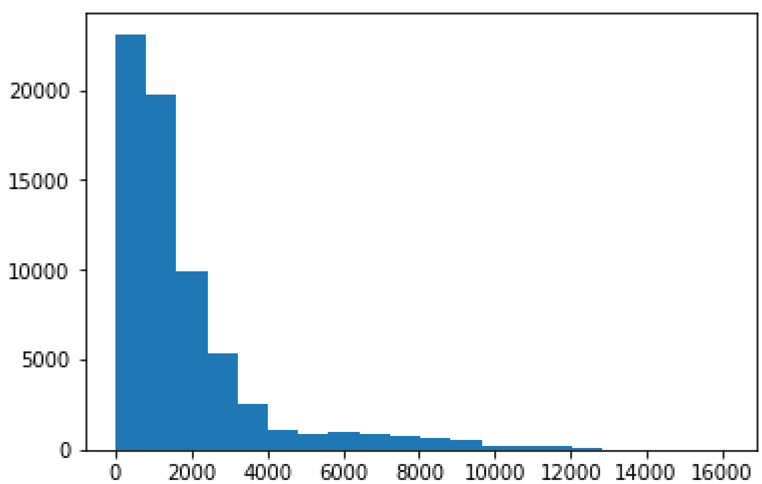
In [32]:
```
#Calculate route lengths
import math

def haversine(lon1, lat1, lon2, lat2):
    # Convert coordinates to floats.
    lon1, lat1, lon2, lat2 = [float(lon1), float(lat1), float(lon2), flo
at(lat2)]
    # Convert to radians from degrees.
    lon1, lat1, lon2, lat2 = map(math.radians, [lon1, lat1, lon2, lat2])
    # Compute distance.
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = math.sin(dlat/2)**2 + math.cos(lat1) * math.cos(lat2) * math.sin
(dlon/2)**2
    c = 2 * math.asin(math.sqrt(a))
    km = 6367 * c
    return km
```

In [33]:
```
#Calculate distance per flight
def calc_dist(row):
    dist = 0
    try:
        # Match source and destination to get coordinates.
        source = airports[airports["id"] == row["source_id"]].iloc[0]
        dest = airports[airports["id"] == row["dest_id"]].iloc[0]
        # Use coordinates to compute distance.
        dist = haversine(dest["longitude"], dest["latitude"], source["lo
ngitude"], source["latitude"])
    except (ValueError, IndexError):
        pass
    return dist
```

In [34]:
```
route_lengths = routes.apply(calc_dist, axis=1)
```

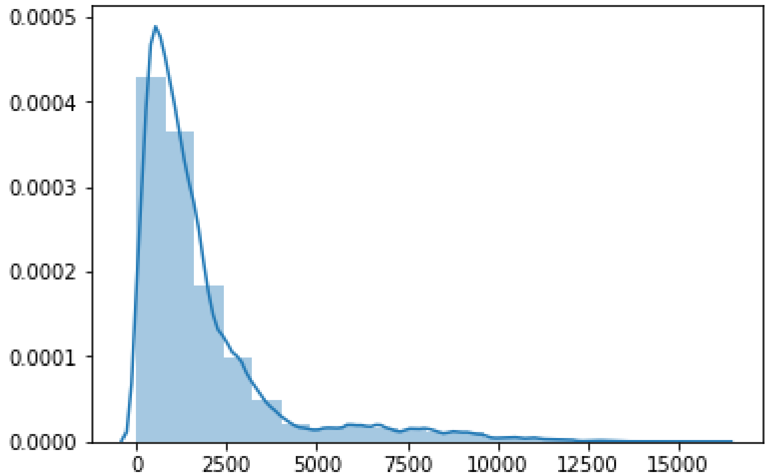
In [35]:
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.hist(route_lengths, bins=20)
```

Out[35]:
```
(array([2.3116e+04, 1.9726e+04, 9.9430e+03, 5.2980e+03, 2.5900e+03,
        1.0630e+03, 8.3900e+02, 1.0170e+03, 9.1200e+02, 7.7100e+02,
        6.4900e+02, 5.5300e+02, 2.4700e+02, 2.3200e+02, 1.4600e+02,
        4.4000e+01, 3.2000e+01, 2.0000e+00, 0.0000e+00, 4.0000e+00]),
 array([    0.        ,   803.60790188,  1607.21580375,  2410.82370563,
         3214.43160751,  4018.03950938,  4821.64741126,  5625.25531313,
         6428.86321501,  7232.47111689,  8036.07901876,  8839.68692064,
         9643.29482252, 10446.90272439, 11250.51062627, 12054.11852815,
        12857.72643002, 13661.3343319 , 14464.94223378, 15268.55013565,
        16072.15803753]),
 <a list of 20 Patch objects>)
```



In [36]:
```
import seaborn
seaborn.distplot(route_lengths, bins=20)
```

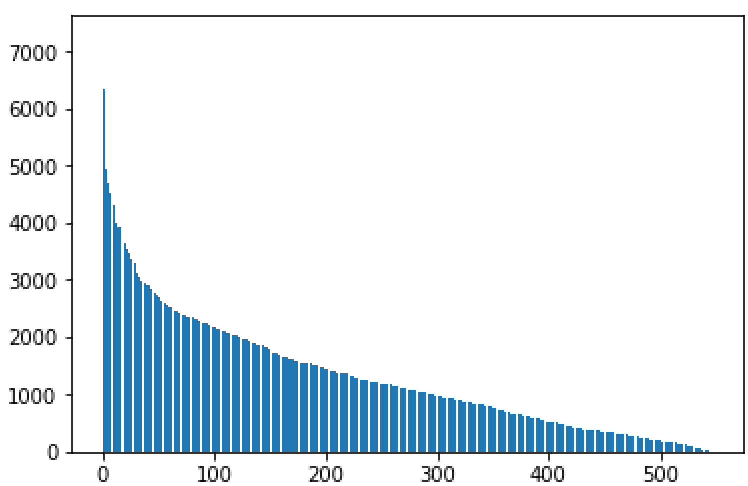
Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x12853cf28>



In [37]:
```
import numpy

# Put relevant columns into a dataframe.
route_length_df = pd.DataFrame({"length": route_lengths, "id": routes["a
irline_id"]})
# Compute the mean route length per airline.
airline_route_lengths = route_length_df.groupby("id").aggregate(numpy.me
an)
# Sort by length so we can make a better chart.
airline_route_lengths = airline_route_lengths.sort_values("length", asce
nding=False)
```

In [38]:
```
plt.bar(range(airline_route_lengths.shape[0]), airline_route_lengths["le
ngth"])
```

Out[38]: <BarContainer object of 547 artists>

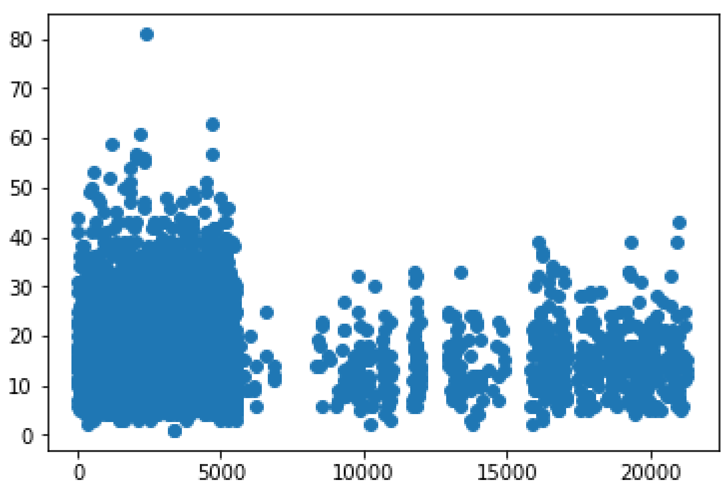


In [39]:
```
def lookup_name(row):
    try:
        # Match the row id to the id in the airlines dataframe so we can
 get the name.
        name = airlines["name"][airlines["id"] == row["id"]].iloc[0]
    except (ValueError, IndexError):
        name = ""
    return name

# Add the index (the airline ids) as a column.
airline_route_lengths["id"] = airline_route_lengths.index.copy()
# Find all the airline names.
airline_route_lengths["name"] = airline_route_lengths.apply(lookup_name,
 axis=1)
# Remove duplicate values in the index.
airline_route_lengths.index = range(airline_route_lengths.shape[0])
```

In [40]:
```
long_routes = len([k for k in route_lengths if k > 10000]) / len(route_l
engths)
medium_routes = len([k for k in route_lengths if k < 10000 and k > 2000
]) / len(route_lengths)
short_routes = len([k for k in route_lengths if k < 2000]) / len(route_l
engths)
```

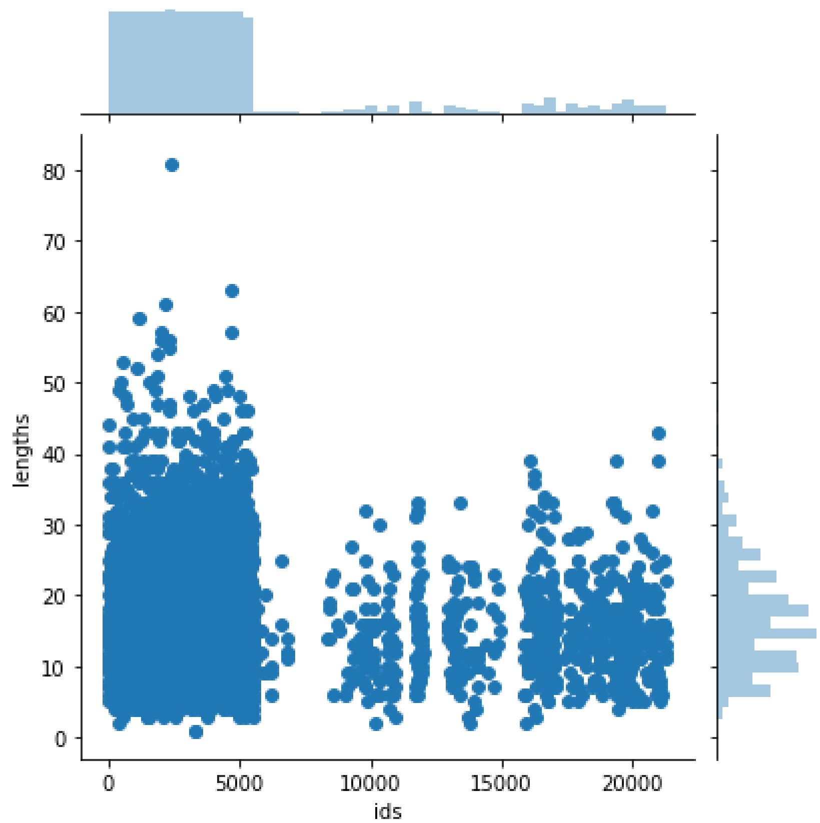
In [41]:
```
name_lengths = airlines["name"].apply(lambda x: len(str(x)))
plt.scatter(airlines["id"].astype(int), name_lengths)
```

Out[41]: <matplotlib.collections.PathCollection at 0x12907f1d0>



In [42]:
```
data = pd.DataFrame({"lengths": name_lengths, "ids": airlines["id"].asty
pe(int)})
seaborn.jointplot(x="ids", y="lengths", data=data)
```

Out[42]: <seaborn.axisgrid.JointGrid at 0x1286701d0>



Question 1

Question - 1

Suppose $Z \sim N(\mu, \sigma^2)$

n samples from Z $[z_1, z_2 \cdots z_n]$.

Let

$$Z_{avg} = \sum_{i=1}^{n} \frac{Z_i}{n} \rightarrow \text{mean}$$

Now consider,

$$P(Z_{avg} > a).$$

By central limit theorem,

$$P\left(\frac{Z_{avg} - \mu}{\sigma(z)} > \frac{a-\mu}{\sigma(z)}\right).$$

$$var(z) = \sum var\left(\frac{x_i}{n}\right)$$

$$\boxed{var(ax) = a^2 var(x)} \qquad = \frac{1}{n^2} var \sum(x_i)$$

$$= \frac{var(x_i)}{n}$$

$$\sigma(z) = \frac{1}{\sqrt{n}} \sigma(x_i).$$

(a) $$P\left(\frac{Z_{avg} - \mu}{\sigma(z)} > \frac{a-\mu}{\sigma(z)}\right)$$

$$P\left(\frac{Z_{avg}-\mu}{\sigma(z)} \le \frac{0.1-0}{1/\sqrt{10^4}}\right)$$

$$1-\phi(10) = \underline{\underline{0}}$$

(b) $P(Z_{avg} > 0.01)$

$$P\left(\frac{Z_{avg}-\mu}{\sigma(z)} \le \frac{0.01-0}{1/100}\right)$$

$$= 1-\phi(1) \approx \underline{\underline{0.158}}$$

(c) $P(Z_{avg} > 0.001)$

$$P\left(\frac{Z_{avg}-\mu}{\sigma(z)} \le \frac{0.001-0}{1/100}\right)$$

$$= 1-\phi(0.1) \approx \underline{\underline{0.46}}.$$

Part 2

$$P(Z_{mean} > A) = 1-\phi\left(\frac{A-\mu}{\sqrt{\sigma^2/n}}\right).$$

$$\therefore P(Z_{mean}-\mu > n^{-1/3}) = P(Z_{mean} > n^{-1/3}+\mu)$$

$$= 1 - \phi\left(\frac{n^{-1/3} + \mu - \mu}{\sqrt{\sigma^2/n}}\right)$$

$$= 1 - \phi\left(\frac{n^{1/6}}{\sigma}\right)$$

(b) $P\left(Z_{mean} - \mu > n^{-1/2}\right) = P\left(Z_{mean} > n^{-1/2} + \mu\right)$

$$= 1 - \phi\left(\frac{n^{-1/2} + \mu - \mu}{\sqrt{\sigma^2/n}}\right)$$

$$= 1 - \phi\left(\frac{1}{\sigma}\right)$$

(c) $P\left(Z_{mean} - \mu > n^{-2/3}\right) = P\left(Z_{mean} > n^{-2/3} + \mu\right)$

$$= 1 - \phi\left(\frac{n^{-2/3} + \mu - \mu}{\sqrt{\sigma^2/n}}\right)$$

$$= 1 - \phi\left(\frac{n^{-1/6}}{\sigma}\right)$$

Question 2

Question - 2

$$h(x) = x \cdot \beta$$

True model according to which data are generated is :-

$$y_i = x_i \cdot \beta + e_i \rightarrow \text{noise}.$$

$$y_i, x_i, \beta, e_i \in \mathbb{R}.$$

(a) $$\min_{\beta} = \frac{1}{n} \sum_{i=1}^{n} (x_i \beta - y_i)^2.$$

To prove,

$$A\beta^2 + B\beta + C.$$

Using Euclidean norm,

$$\min_{\beta} = \frac{1}{n} \| (X\beta - Y) \|^2.$$

$$= \frac{1}{n} (X\beta - Y)^T (X\beta - Y).$$

$$= \frac{1}{n} \left(X^T \beta^T - Y^T \right) \left(X\beta - Y \right)$$

$$= \frac{1}{n} \left(X^T X \beta \beta^T - \beta^T X^T Y - Y^T X \beta + Y^T Y \right)$$

$$= \frac{1}{n} \left(\beta^T X^T X \beta - 2\beta^T X^T Y + Y^T Y \right).$$

$$= \frac{1}{n}\left(\beta^2 XX^T - 2\beta^T X^T Y + YY^T\right) \rightarrow$$ Let this be equation 1.

$A = \frac{XX^T}{n}$; $B = -2$; $C = \frac{YY^T}{n}$

(b) Show $A \geq 0$,

Case 1 : When X is +ve
Case 2 : When X is -ve
} The square is always positive

$\therefore$ $\underline{A \geq 0}$.

(c) From (a) we know.

$$\min_{\beta} : \frac{1}{n}\left[\beta^T X^T X \beta - 2\beta^T X^T Y - Y^T Y\right].$$

Take the gradient,

Fact : $\nabla\left(\beta^T X^T X \beta\right) = 2X^T X\beta$

$\nabla\left(\beta^T X^T Y\right) = X^T Y$.

Applying fact to,

$$\min_{\beta} : \frac{1}{n}\left(\beta^T X^T \beta - 2\beta^T X^T Y + Y^T Y\right)$$

$$\nabla(\ ) = 0$$

$$\frac{1}{\cancel{n}}\left(\cancel{2}X^T X\beta - \cancel{2}X^T Y + 0\right) = 0$$

5

$$(X^T X)\beta = X^T y$$

$$\boxed{\hat{\beta} = (X^T X)^{-1} X^T y}$$

(d) $\hat{\beta} = (X^T X)^{-1} X^T y$

given $y_i = x_i \beta + e_i$

$$\hat{\beta} = (X^T X)^{-1} X^T (X\beta + e)$$

$$= (X^T X)^{-1} X^T X \beta + (X^T X)^{-1} X^T e.$$

$$\hat{\beta} = \beta + (X^T X)^{-1} X^T e$$

$$\hat{\beta} = \beta + Ze.$$

$$\therefore \quad Z = (X^T X)^{-1} X^T$$