Lab 2

September 20, 2018

1 Lab 2

Aimun Khan

aak2629

Caleb Johnson

CDJ2273

2 Problem 1

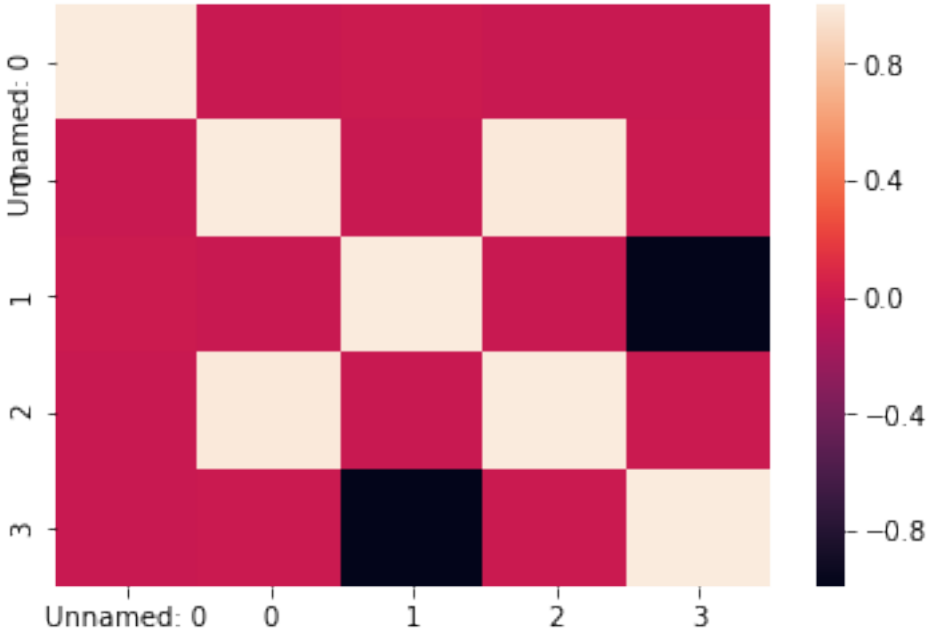
2.0.1 Part a.

In [4]:
```
#1a.
import pandas
import seaborn
import matplotlib.pyplot as plt

DF1 = pandas.read_csv("Lab2_Data/DF1")
print(DF1)
print(DF1.corr())

plot = seaborn.heatmap(DF1.corr())
plt.show()
print("0 and 2 are correlated with correlation coefficient 0.990066")
print("1 and 3 are correlated with correlation coefficient -0.990235")
```

```
      Unnamed: 0         0         1         2         3
0              0  1.038502  0.899865  0.835053 -0.971528
1              1  0.320455 -0.647459  0.149079  0.352593
2              2  0.055480  2.234771  0.271672 -2.108739
3              3 -0.007260 -0.524299 -0.126550  0.670827
4              4 -1.237390 -1.377017 -1.049932  1.342079
5              5  0.477841  0.032660  0.336723 -0.171675
6              6 -0.486923 -1.128336 -0.459850  1.113013
7              7  0.313020  0.677323  0.123082 -0.617958
```

8	8	0.919790	-0.539665	0.956577	0.821389
9	9	0.574238	-1.024339	0.471622	1.006623
10	10	-0.745211	1.117401	-0.955933	-1.128786
11	11	-0.472249	1.819872	-0.660452	-1.782977
12	12	0.426001	1.501646	0.275335	-1.466056
13	13	1.529169	1.964452	1.485045	-1.950166
14	14	0.454290	-0.643795	0.417083	0.623628
15	15	2.225789	-0.015177	2.123942	0.125185
16	16	0.325455	-0.679482	0.589621	0.849904
17	17	-0.620078	-0.260013	-0.631947	0.314077
18	18	-0.968355	-0.576313	-0.852067	0.608772
19	19	-0.497867	-0.826118	-0.474271	0.877328
20	20	0.138424	2.346433	0.246833	-2.328777
21	21	-0.612432	0.867661	-0.426946	-1.218642
22	22	-3.201179	-0.295933	-3.000997	0.163514
23	23	-0.703764	-0.895308	-0.399714	0.728313
24	24	1.601584	-0.255763	1.724664	0.133922
25	25	0.191843	-0.728643	0.266744	0.769094
26	26	0.125723	0.471644	0.053597	-0.459898
27	27	0.221168	0.606079	0.386691	-0.897673
28	28	-0.231375	-2.708069	-0.003476	2.633477
29	29	0.536865	-0.276776	0.394512	0.031890
...	...	...	...	...	...
9970	9970	-2.730439	-1.547178	-2.701821	1.708031
9971	9971	0.688554	-0.370272	0.725510	0.236869
9972	9972	-0.884338	0.889492	-0.785772	-0.823868
9973	9973	-2.400426	0.117176	-2.372839	-0.025592
9974	9974	-1.598474	0.268953	-1.623537	-0.362582
9975	9975	-0.535834	0.012192	-0.441412	0.095823
9976	9976	-1.100836	2.479797	-1.014950	-2.628093
9977	9977	-0.008764	-0.075486	0.009927	-0.219688
9978	9978	0.683233	-0.140395	0.633649	-0.088128
9979	9979	-1.406525	-0.908348	-1.271741	0.882756
9980	9980	-0.173609	-0.185395	-0.343300	0.256917
9981	9981	0.960713	-0.128936	0.722877	0.139783
9982	9982	-0.226978	1.026923	-0.154546	-1.041752
9983	9983	0.550183	1.805770	0.256797	-1.447349
9984	9984	0.447656	-0.905489	0.640393	0.796162
9985	9985	0.165993	0.222795	0.151294	-0.023453
9986	9986	-0.814842	-1.089027	-0.902200	1.057734
9987	9987	0.207744	1.102626	0.116032	-1.273074
9988	9988	-0.452244	0.129441	-0.517421	-0.328953
9989	9989	-0.493284	0.222432	-0.717655	-0.267048
9990	9990	-0.423431	-0.415418	-0.465615	0.494101
9991	9991	-1.684161	-0.182262	-1.538363	0.004538
9992	9992	-1.780461	0.910024	-1.597428	-0.772670
9993	9993	-0.941366	0.055372	-0.657820	0.192450
9994	9994	0.790076	1.339041	0.717082	-1.282713

```
9995        9995 -0.632309 -0.145873 -0.797517  0.436184
9996        9996  0.679417 -0.530216  0.526470  0.439397
9997        9997  0.890697 -2.210855  1.072751  2.285372
9998        9998  0.475293  0.490971  0.536909 -0.195772
9999        9999  1.207406  0.819239  1.230797 -0.752397

[10000 rows x 5 columns]
            Unnamed: 0         0         1         2         3
Unnamed: 0    1.000000 -0.003991  0.008789 -0.004044 -0.007086
0            -0.003991  1.000000 -0.003998  0.990066  0.004111
1             0.008789 -0.003998  1.000000 -0.004085 -0.990235
2            -0.004044  0.990066 -0.004085  1.000000  0.004067
3            -0.007086  0.004111 -0.990235  0.004067  1.000000
```




```
0 and 2 are correlated with correlation coefficient 0.990066
1 and 3 are correlated with correlation coefficient -0.990235
```

2.0.2 Part b.

```
In [5]: #1b.
        import numpy
        print(numpy.cov(DF1.drop(columns=["Unnamed: 0"]).corr()))
```

```
[[ 0.33002024 -0.00564931  0.33000902  0.00242671]
 [-0.00564931  0.66019896 -0.00566404 -0.66017005]
```

```
 [ 0.33000902 -0.00566404  0.33006359  0.0024408 ]
 [ 0.00242671 -0.66017005  0.0024408   0.66017262]]
```

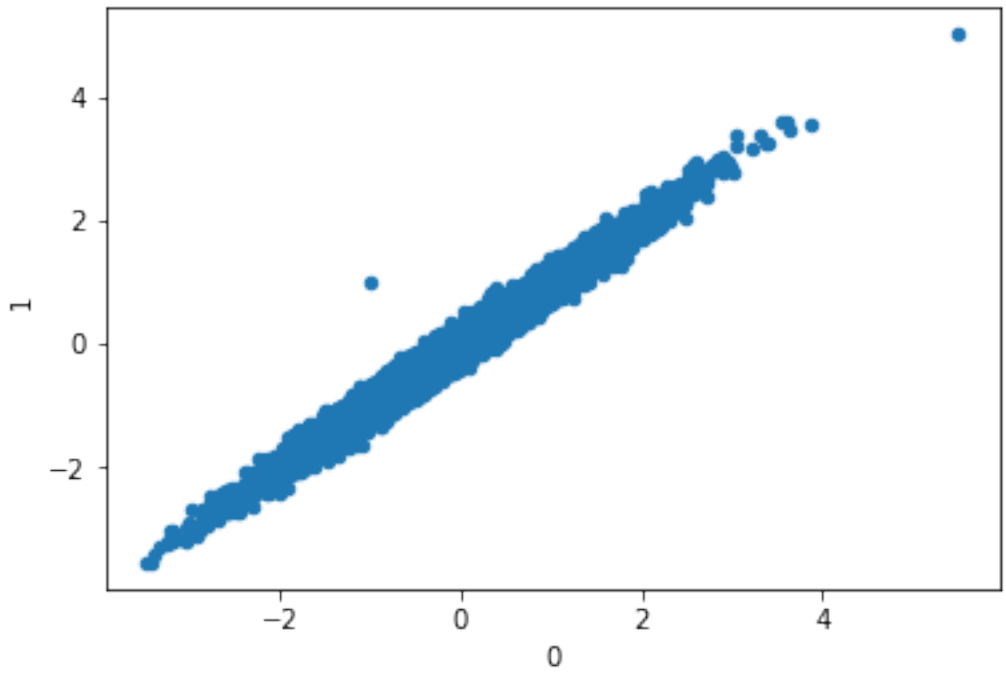
2.0.3 Part c.

In [6]:
```
#1c.
```

3 Problem 2

In [7]:
```
#2.
import numpy
import pandas
import seaborn
import matplotlib.pyplot as plt

# Find pairwise correlations
DF2 = pandas.read_csv("Lab2_Data/DF2")
print(list(DF2))
DF2.plot.scatter('0','1')
plt.show()

print("Two outliers, (-1,1) looks like more of an outlier because its far away from the

data_matrix = DF2.as_matrix()

cov = DF2.cov()
cov = np.linalg.inv(cov)
data = cov.dot(data_matrix.T)

dataT = data.T
x = dataT[:, 1]
y = dataT[:, 2]

plt.scatter(x, y)
plt.show()

print("We've isolated only the point (0,1) as an outlier by inverting the matrix")
```

```
['Unnamed: 0', '0', '1']
```




```
Two outliers, (-1,1) looks like more of an outlier because its far away from the trend line, bu
```

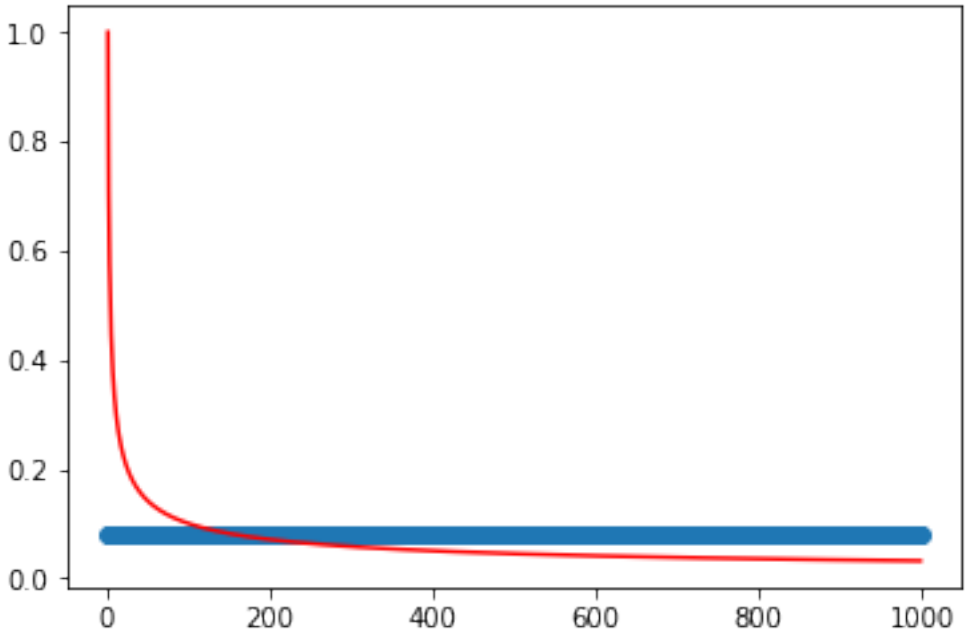
```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-7-12dcf730cc71> in <module>()
         16
         17 cov = DF2.cov()
    ---> 18 cov = np.linalg.inv(cov)
         19 data = cov.dot(data_matrix.T)
         20


        NameError: name 'np' is not defined
```

4 Problem 3

```
In [8]: #pt 2
        print("Pt. 1:")
        import numpy as np
```

```
        b_hats = []
        for x in range (0, 1000):
            n = 150
            x = np.random.normal(0, 1, n)
            e = np.random.normal(0, 1, n)
            b_hat = (x.dot(e)) / (x.dot(x))
            b_hats.append(b_hat)

        print("Mean:", np.mean(b_hats))
        print("Std:", np.std(b_hats))
        print("B = 0.15 is significant")

        #pt 2
        print("\nPt. 2:")
        n = 1000

        x_axis = []
        x_axis2 = []
        y_axis = []
        for value in range (1, n):
            x = np.random.normal(0, 1, 150)
            e = np.random.normal(0, 1, 150)
            b_hat = (x.dot(e)) / (x.dot(x))
            b_hats.append(b_hat)

            x_axis.append(value)
            x_axis2.append(1 / (int(value)**.5))
            y_axis.append(np.std(b_hats))

        plt.scatter(x_axis, y_axis)
        plt.plot(x_axis, x_axis2, 'r')
        plt.show()


        print("Good fit to line")
        print("Mean:", np.mean(b_hats))
        print("Std:", np.std(b_hats))
```

```
Pt. 1:
Mean: 0.0025935168950072737
Std: 0.08109094686948413
B = 0.15 is significant

Pt. 2:
```




```
Good fit to line
Mean: -0.00038289061621863584
Std: 0.08126770366075596
```

5 Problem 4

Problem 4)

6 4.a)

```
In [9]: def topNames(k, year):
            data = pandas.read_csv("Names/Names/yob" + year + ".txt")
            k = data.nlargest(k, '52667')
            return k['Jessica']
```

7 4.b)

```
In [12]: def nameFrequency(name, y1, y2):
             males = 0
             females = 0


             for i in range(y1,y2):
```

```
            data = pandas.read_csv("Names/Names/yob" + str(i) + ".txt")
            data.columns = ['Name', 'Sex', 'Number']
            data = data.set_index(['Name'])
            a = data.loc[data.index.isin([name])]
            a = pandas.DataFrame(data = a)
            a.columns = ["Sex", "Number"]

            sex = a["Sex"]
            amt = a["Number"]
            amt = pandas.DataFrame(data = amt)
            amt.columns = ['Number']
            if sex.size == 0:
                b = {'col': [0, 0]}
                result = pandas.DataFrame(data=b)
            if sex.size == 1:
                if sex[0] == 'F':
                    females += amt.iloc[0]['Number']
                    b = {'col': [[males], [females]]}
                    result = pandas.DataFrame(data = b)
                else:
                    males += amt.iloc[0]['Number']
                    b = {'col': [[males], [females]]}
                    result = pandas.DataFrame(data = b)
            if sex.size == 2:
                males += amt.iloc[1]['Number']
                females += amt.iloc[0]['Number']
                b = {'col1': [[males], [females]]}
                result = pandas.DataFrame(data = b)

        return result
```

```
In [13]: print(nameFrequency('Caleb', 1975, 1985))
```

```
      col1
0  [10578]
1     [55]
```

```
In [16]: import matplotlib.pyplot as plt
         import numpy as np

         %matplotlib inline

         freq = []
         total = 0
         years = [1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984]

         for i in range(1975,1985):
```

```
        freq.append(nameFrequency('Caleb', i, i+1))

    new = []
    for j in range(0,len(freq)):
        new.append(freq[j].iloc[0].item())
        total += sum(new[j])

    total


    #plt.show()
```

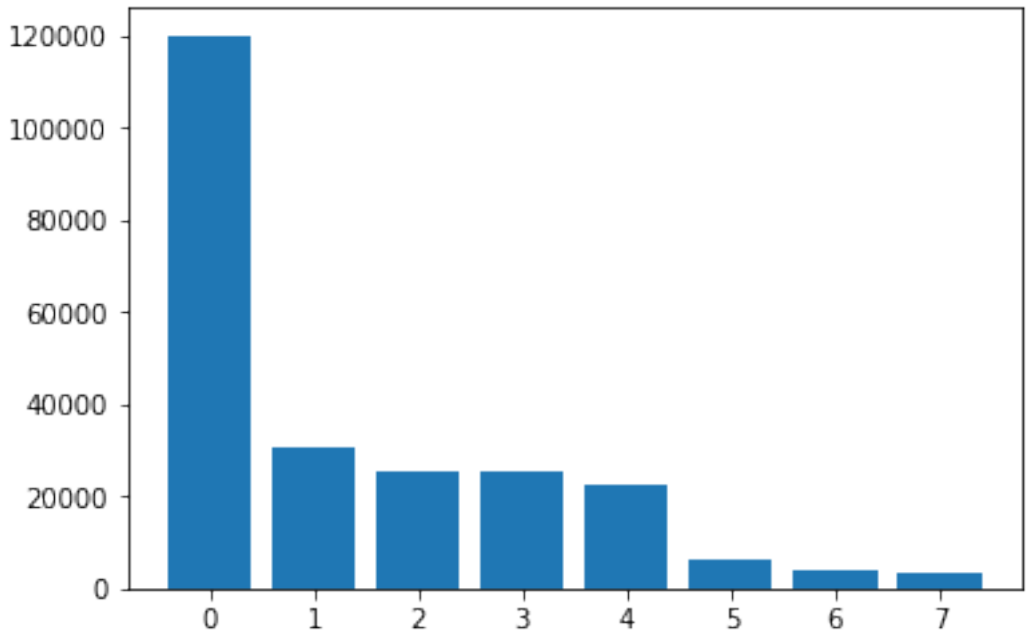
Out[16]: 10578

8 Problem 5

In [18]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

#Print the first 5 rows of the DataFrame
tweets = pd.read_csv("tweets.csv")
tweets.head()

#Create a function that finds what candidate names occur in a piece of text.
#Use the apply method on DataFrames to generate a new column called candidate that
#contains what candidate(s) the tweet mentions.
def get_candidate(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["candidate"] = tweets.apply(get_candidate,axis=1)

#bar plot
print("Bar plot:")
counts = tweets["candidate"].value_counts()
plt.bar(range(len(counts)), counts)
```

```
        plt.show()

        print(counts)
```

Bar plot:



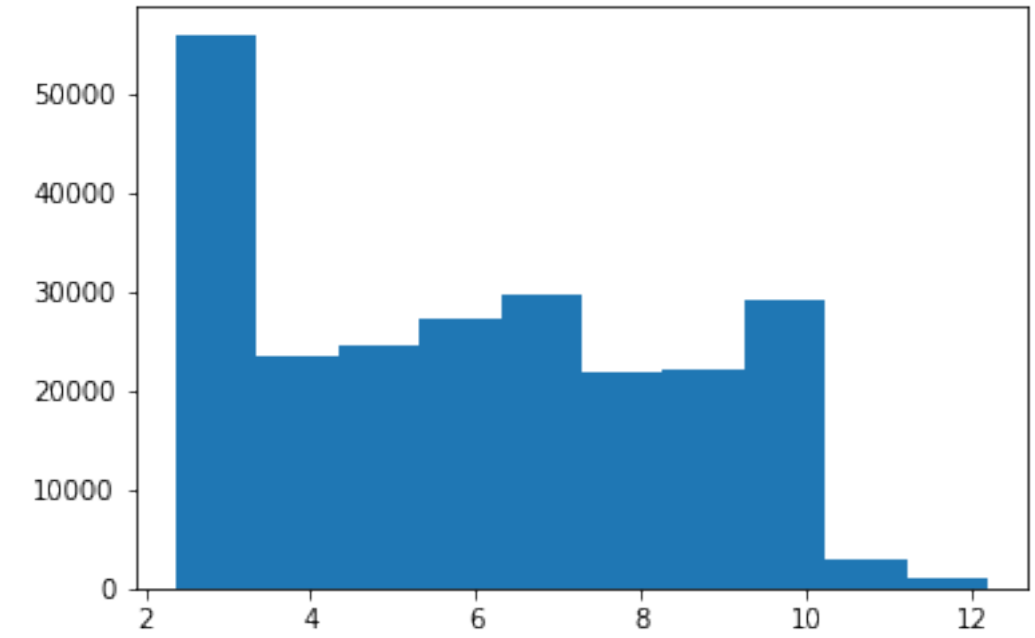

```
trump                     119998
clinton,trump              30521
                           25429
sanders                    25351
clinton                    22746
clinton,sanders             6044
clinton,trump,sanders       4219
trump,sanders               3172
Name: candidate, dtype: int64
```

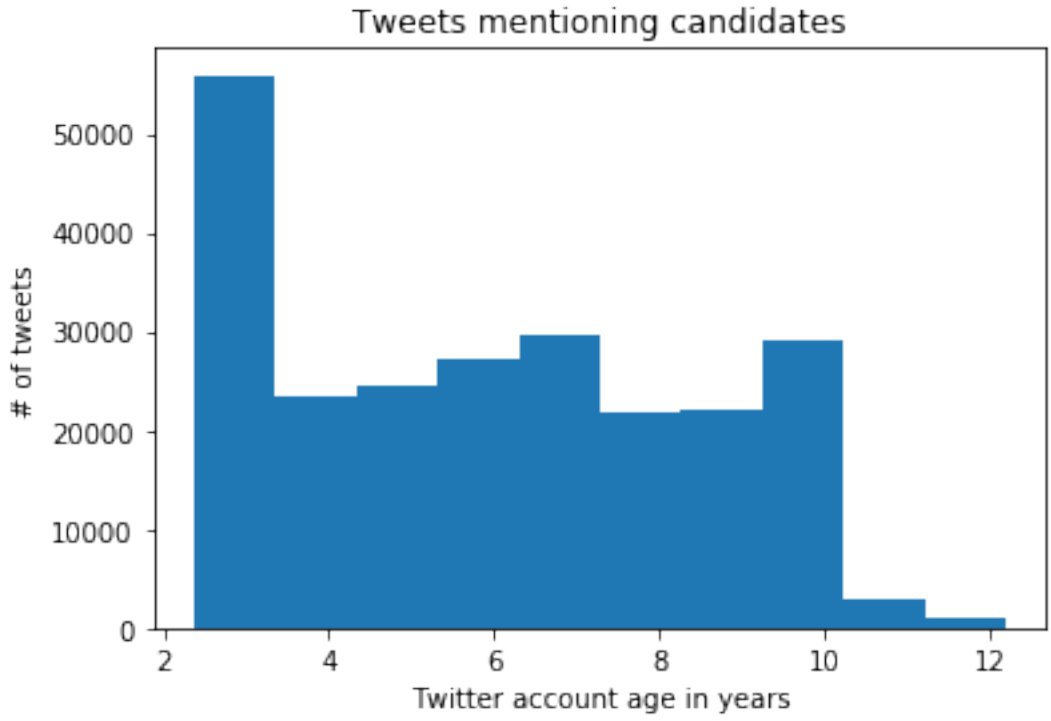
```
In [19]: #histogram
         print("Histogram:")
         from datetime import datetime

         tweets["created"] = pd.to_datetime(tweets["created"])
         tweets["user_created"] = pd.to_datetime(tweets["user_created"])

         tweets["user_age"] = tweets["user_created"].apply(lambda x: (datetime.now() - x).total
```

```
plt.hist(tweets["user_age"])
plt.show()
plt.clf()

#add labels
plt.hist(tweets["user_age"])
plt.title("Tweets mentioning candidates")
plt.xlabel("Twitter account age in years")
plt.ylabel("# of tweets")
plt.show()
plt.clf()

#stacked histogram
print("Stacked:")
cl_tweets = tweets["user_age"][tweets["candidate"] == "clinton"]
sa_tweets = tweets["user_age"][tweets["candidate"] == "sanders"]
tr_tweets = tweets["user_age"][tweets["candidate"] == "trump"]
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.show()
plt.clf()

#annotating the histogram
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=(1, 300), xytext=(2, 300),
             arrowprops=dict(facecolor='black'))
plt.show()
```

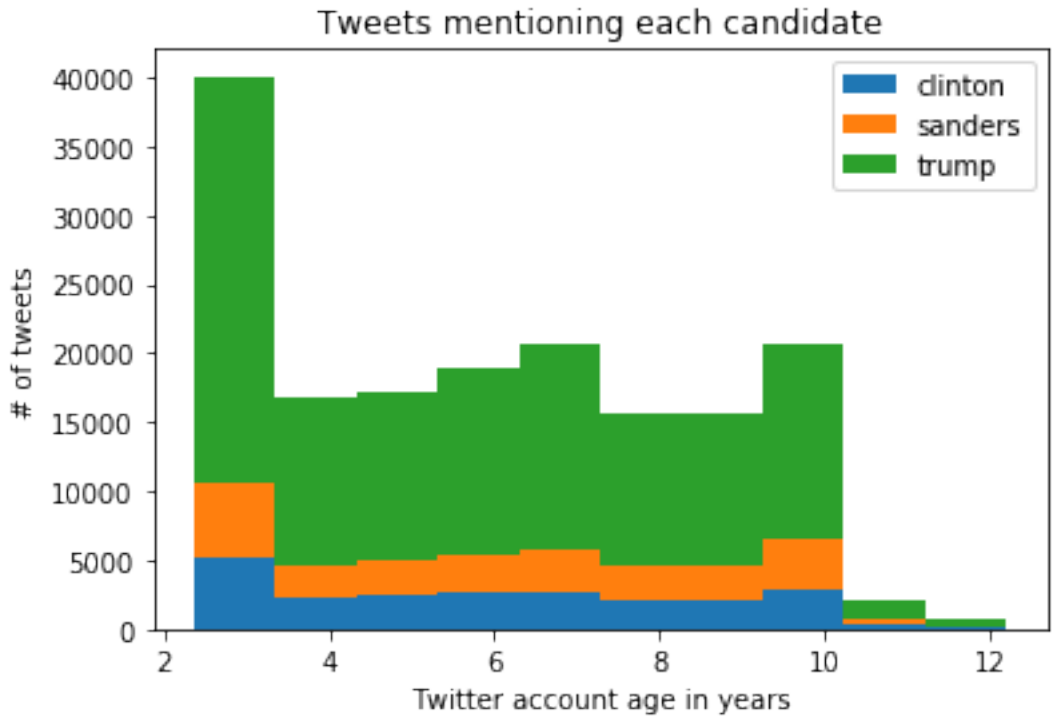
```
plt.clf()
```

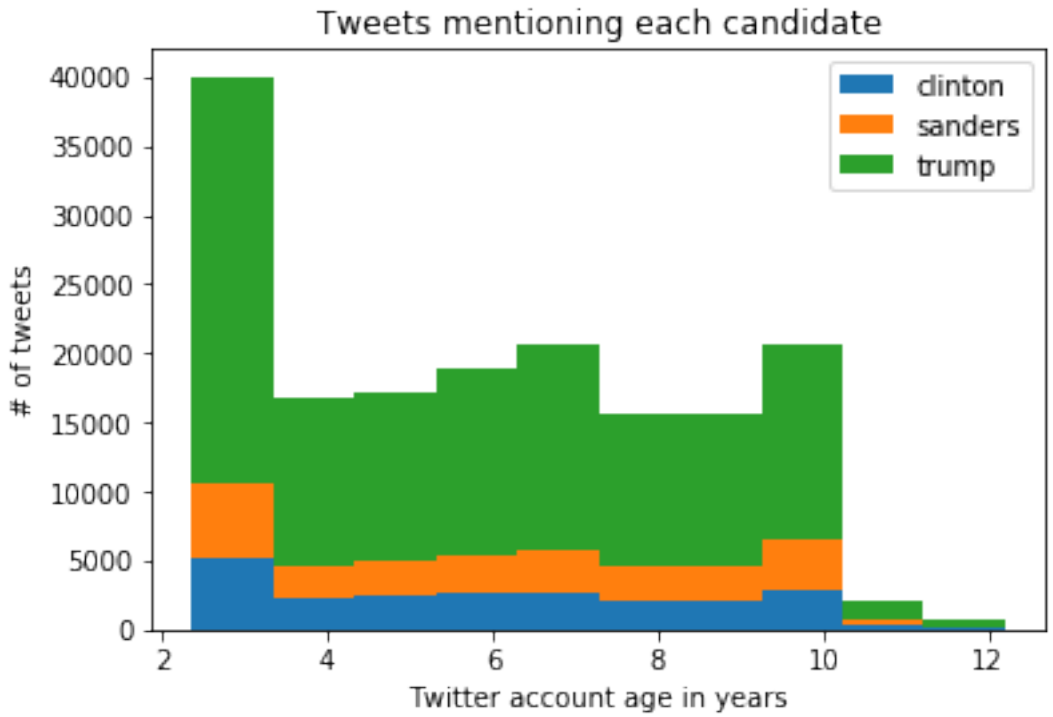
Histogram:





```
Stacked:


C:\Users\caleb\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:52: FutureWarning: reshape
  return getattr(obj, method)(*args, **kwds)
```






```
<matplotlib.figure.Figure at 0x27afcb4f780>
```

```
In [21]: #extracting colors
         print("Colors:")
         import matplotlib.colors as colors

         tweets["red"] = tweets["user_bg_color"].apply(lambda x: colors.hex2color('#{0}'.forma
         tweets["blue"] = tweets["user_bg_color"].apply(lambda x: colors.hex2color('#{0}'.forma

         #creating a plot
         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["candidate"] == "trump"].values)
         ax1.set_title('Red in Trump tweeters')
```

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

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

plt.tight_layout()
plt.show()

#removing common bg colors
print("Removing colors:")
tweets["user_bg_color"].value_counts()
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["candidate"] == "trump"].values)
    ax1.set_title('Red in Trump tweets')

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

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

    plt.tight_layout()
    plt.show()

create_plot(tc)
```

Colors:

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)

~\Anaconda3\lib\site-packages\matplotlib\colors.py in to_rgba(c, alpha)
    131     try:
--> 132         rgba = _colors_full_map.cache[c, alpha]
    133     except (KeyError, TypeError):  # Not in cache, or unhashable.
```

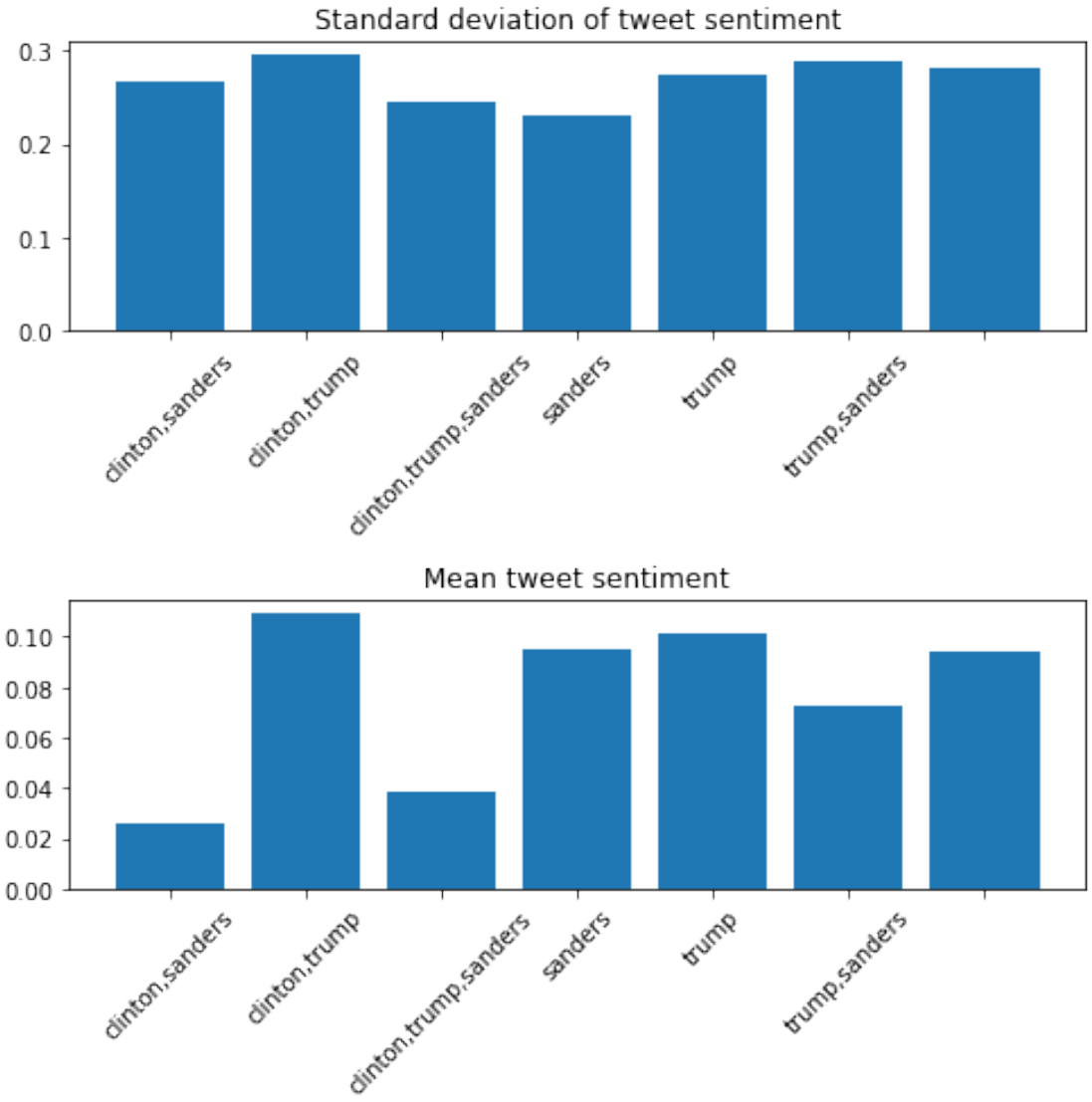
```
    KeyError: ('#22330', None)


During handling of the above exception, another exception occurred:


    ValueError                                Traceback (most recent call last)

    <ipython-input-21-8bff9092768c> in <module>()
      4 import matplotlib.colors as colors
      5
----> 6 tweets["red"] = tweets["user_bg_color"].apply(lambda x: colors.hex2color('#{0}'.for
      7 tweets["blue"] = tweets["user_bg_color"].apply(lambda x: colors.hex2color('#{0}'.fo
      8


    ~\Anaconda3\lib\site-packages\pandas\core\series.py in apply(self, func, convert_dtype
   2549             else:
   2550                 values = self.asobject
-> 2551                 mapped = lib.map_infer(values, f, convert=convert_dtype)
   2552
   2553         if len(mapped) and isinstance(mapped[0], Series):


    pandas/_libs/src/inference.pyx in pandas._libs.lib.map_infer()


    <ipython-input-21-8bff9092768c> in <lambda>(x)
      4 import matplotlib.colors as colors
      5
----> 6 tweets["red"] = tweets["user_bg_color"].apply(lambda x: colors.hex2color('#{0}'.for
      7 tweets["blue"] = tweets["user_bg_color"].apply(lambda x: colors.hex2color('#{0}'.fo
      8


    ~\Anaconda3\lib\site-packages\matplotlib\colors.py in hex2color(c)
    270     Example: #efefef -> (0.93725, 0.93725, 0.93725)
    271     """
--> 272     return ColorConverter.to_rgb(c)
    273
    274


    ~\Anaconda3\lib\site-packages\matplotlib\colors.py in to_rgb(arg)
    303         if *arg* is *RGBA*, the *A* will simply be discarded.
    304         """
--> 305         return to_rgb(arg)
```

```
    306
    307     @staticmethod


~\Anaconda3\lib\site-packages\matplotlib\colors.py in to_rgb(c)
    238     """Convert `c` to an RGB color, silently dropping the alpha channel.
    239     """
--> 240     return to_rgba(c)[:3]
    241
    242


~\Anaconda3\lib\site-packages\matplotlib\colors.py in to_rgba(c, alpha)
    132         rgba = _colors_full_map.cache[c, alpha]
    133     except (KeyError, TypeError):  # Not in cache, or unhashable.
--> 134         rgba = _to_rgba_no_colorcycle(c, alpha)
    135         try:
    136             _colors_full_map.cache[c, alpha] = rgba


~\Anaconda3\lib\site-packages\matplotlib\colors.py in _to_rgba_no_colorcycle(c, alpha)
    176         except ValueError:
    177             pass
--> 178         raise ValueError("Invalid RGBA argument: {!r}".format(orig_c))
    179     # tuple color.
    180     c = np.array(c)


ValueError: Invalid RGBA argument: '#22330'
```

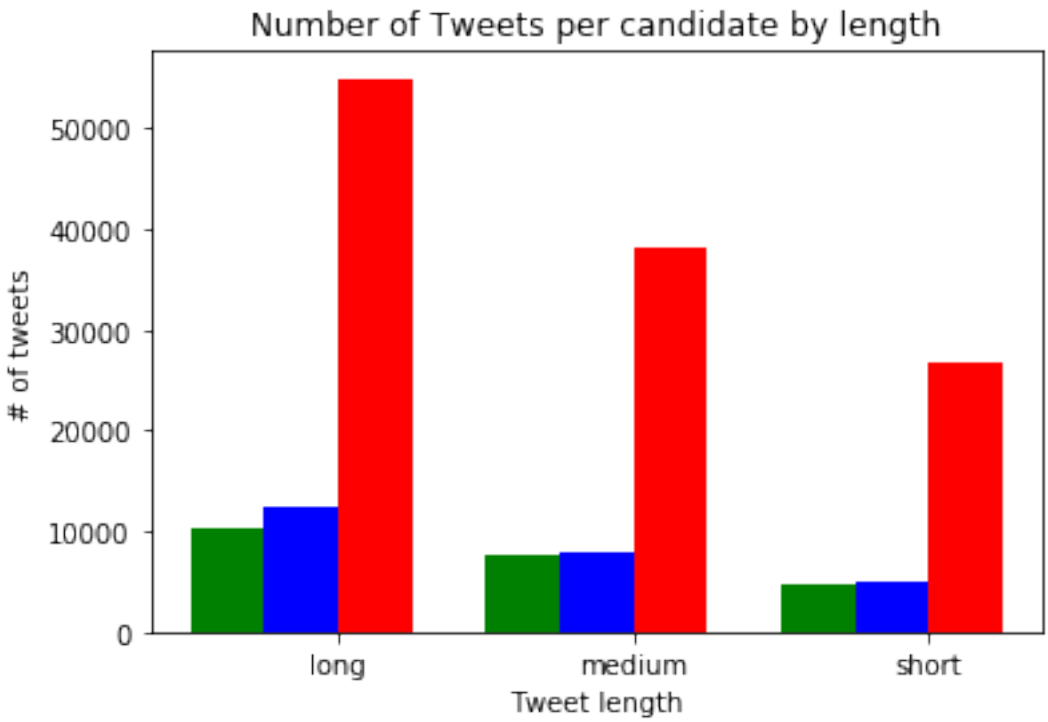
In [22]:
```
#plotting sentiment
print("Plotting sentiment:")
gr = tweets.groupby("candidate").agg([np.mean, np.std])

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()

        #side-by-side bar plot
        print("Side-by-side bar plot:")
        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["candidate"] == candidate].value_cou

        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()
```

Plotting sentiment:



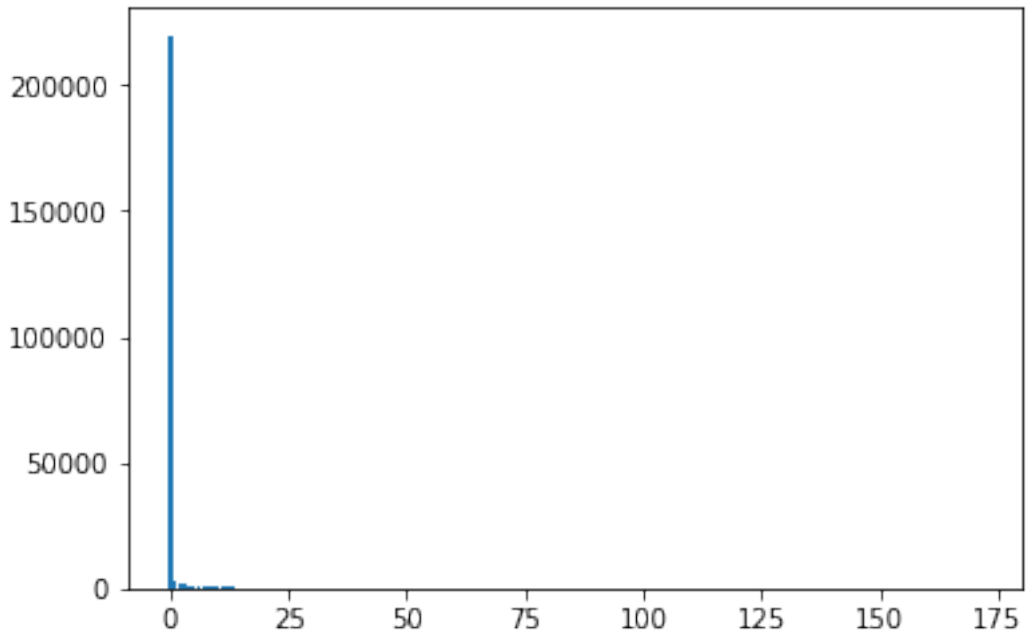
Side-by-side bar plot:




```
In [23]: import pandas as pd
         import matplotlib.pyplot as plt
         import numpy as np
         %matplotlib inline

         #Print the first 5 rows of the DataFrame
         tweets = pd.read_csv("tweets.csv")
         tweets.head()


         #Create a function that finds what candidate names occur in a piece of text.
         #Use the apply method on DataFrames to generate a new column called candidate that
         #contains what candidate(s) the tweet mentions.
         def get_candidate(row):
             candidates = []
             text = row["text"].lower()
             states = [ "Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado",
             for state in states:
                 if state.lower() in text:
                     candidates.append(state)
             return ",".join(candidates)

         tweets["candidate"] = tweets.apply(get_candidate,axis=1)
```

```
#bar plot
print("Bar plot:")
counts = tweets["candidate"].value_counts()
plt.bar(range(len(counts)), counts)
plt.show()

print(counts)
```

```
Bar plot:
```




```
                                                  219747
California                                          3027
Virginia,West Virginia                              2158
Washington                                          1654
Nebraska                                            1271
Virginia                                            1249
Ohio                                                1110
Nebraska,Virginia,West Virginia                      828
New Mexico                                           592
Florida                                              551
Florida,Ohio,Pennsylvania                            483
Nebraska,Virginia                                    423
New York                                             420
Pennsylvania                                         362
Texas                                                306
```

```
Alaska                                                    227
Oregon                                                    216
Iowa                                                      194
Kentucky                                                  178
Louisiana                                                 151
Indiana                                                   136
Utah                                                      130
Maine                                                     115
New Jersey                                                113
Virginia,Washington,West Virginia                         104
Arizona                                                    85
Georgia                                                    72
Florida,Ohio                                               71
Wisconsin                                                  69
Vermont                                                    60
                                                          ...
Nebraska,Texas,Virginia,West Virginia                       1
Kentucky,Ohio                                               1
Alaska,Virginia,West Virginia                               1
Nebraska,Wyoming                                            1
Florida,Washington                                          1
Pennsylvania,Texas                                          1
California,Kentucky,Nebraska,Virginia                       1
Kansas,Oklahoma,Texas                                       1
Montana,Oregon                                              1
New York,Ohio                                               1
California,New Jersey,Oregon,Virginia,West Virginia         1
Florida,Michigan,Ohio,Pennsylvania                          1
Colorado,Nevada                                             1
California,Ohio                                             1
Iowa,Virginia,Washington,West Virginia                      1
Idaho,Utah,Wyoming                                          1
Maryland,Utah                                               1
Florida,New York,Ohio                                       1
Nebraska,Texas                                              1
Ohio,Virginia                                               1
Massachusetts,Michigan                                      1
California,Michigan                                         1
Georgia,Mississippi,Utah                                    1
Pennsylvania,Virginia,West Virginia                         1
California,Georgia                                          1
California,Montana,Oregon                                   1
Arizona,Ohio                                                1
Iowa,Ohio                                                   1
California,New York,Pennsylvania                            1
Arizona,Georgia,Mississippi,North Carolina                  1
Name: candidate, Length: 172, dtype: int64
```

1.a) WE HAVE: $Z \sim N(\mu, \sigma^2)$, Z IS A UNIVARIATE GAUSSIAN.

AND

$Z_{avg} = \sum_{i=1}^{n} \frac{Z_i}{n}$. TO FIND HOW CLOSE Z_{avg} IS TO μ, WE SET

$n = 10{,}000$, $\mu = \emptyset$, $\sigma^2 = 1$. CLT SAYS $P(Z_{avg} > .1) = P\left(\frac{Z_{avg}\sqrt{n}}{\sigma} > .1\frac{\sqrt{n}}{\sigma}\right)$

$= P\left(\frac{Z_{avg}(100)}{1} > 10\right) = P(Z_{avg}^* > .1) = 1 - \Phi(.1) = .46$

$\approx 1 - \Phi(.01) = .496$

$\approx 1 - \Phi(.001) = .499$

1.b) USING CLT AS WE DID ABOVE, WE HAVE:

$$\left|(Z_{avg} - \mu)\right| = \left|\frac{Z_1 + Z_2 + \ldots Z_n - (n)\mu}{\sqrt{n}\sigma}\right|$$

SO

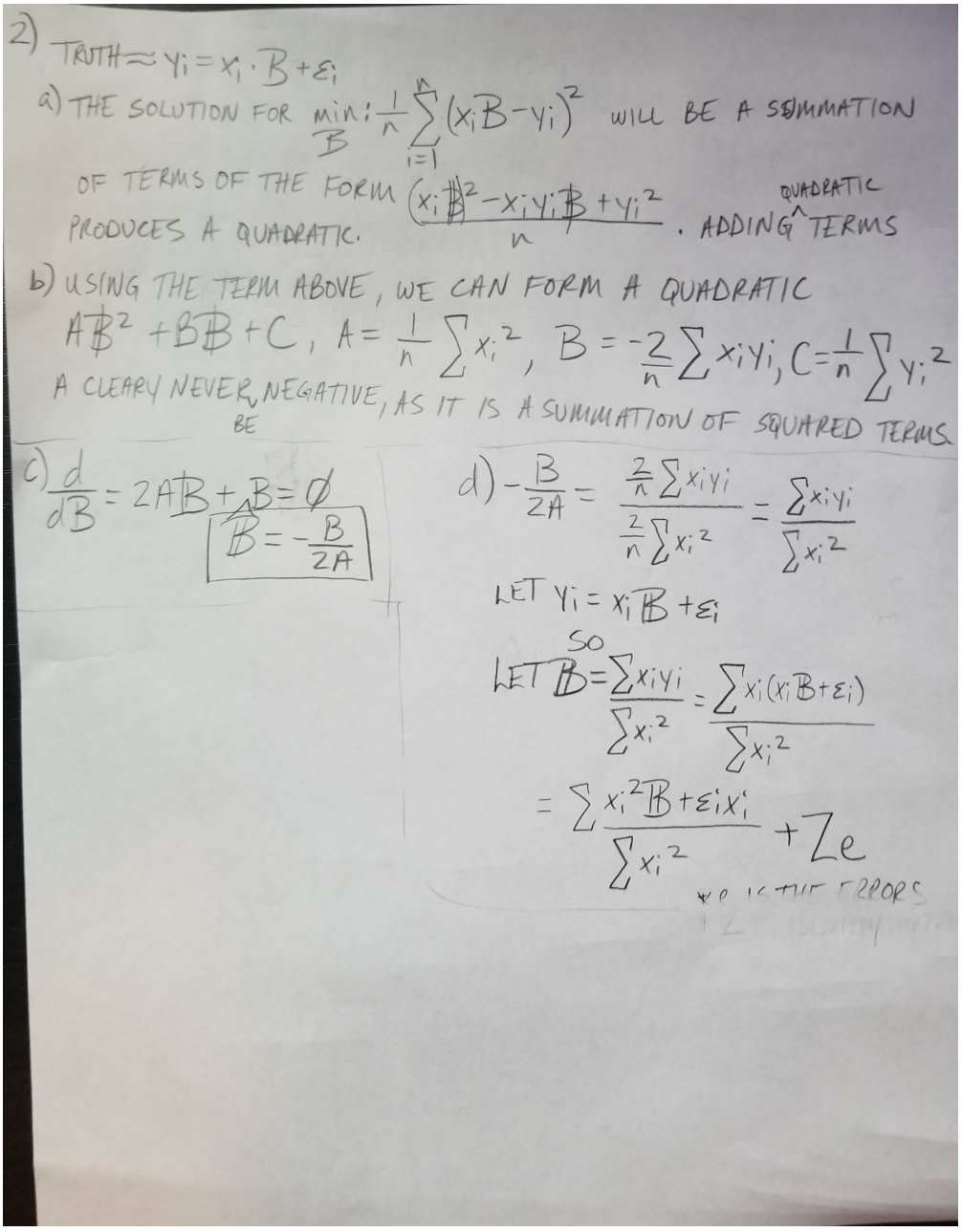
$$P\left(\left|\frac{Z_1 + Z_2 + \ldots Z_n - (n)\mu}{\sqrt{n}\sigma}\right| \geq \left(n^{-1/3}\right)\frac{\sqrt{n}}{\sigma}\right) = 1 - \Phi\left(\frac{\sqrt[6]{n}}{\sigma}\right)$$

OR

$$\ldots \left(n^{-1/2}\right)\frac{\sqrt{n}}{\sigma}\right) = 1 - \Phi\left(\frac{1}{\sigma}\right)$$

OR

$$\ldots \left(n^{-2/3}\right)\frac{\sqrt{n}}{\sigma}\right) = 1 - \Phi\left(\frac{1}{\sqrt[6]{n}\sigma}\right)$$


2) TRUTH $\approx y_i = x_i \cdot \beta + \varepsilon_i$
a) THE SOLUTION FOR $\min_{\beta} \frac{1}{n}\sum_{i=1}^{n}(x_i\beta - y_i)^2$ WILL BE A SUMMATION
OF TERMS OF THE FORM $\frac{x_i^2\beta^2 - x_i y_i \beta + y_i^2}{n}$. ADDING QUADRATIC TERMS
PRODUCES A QUADRATIC.
b) USING THE TERM ABOVE, WE CAN FORM A QUADRATIC
$A\beta^2 + B\beta + C$, $A = \frac{1}{n}\sum x_i^2$, $B = -\frac{2}{n}\sum x_i y_i$, $C = \frac{1}{n}\sum y_i^2$
A CLEARY NEVER NEGATIVE, AS IT IS A SUMMATION OF SQUARED TERMS
BE
c) $\frac{d}{d\beta} = 2A\beta + B = 0$
$\beta = -\frac{B}{2A}$
d) $-\frac{B}{2A} = \frac{\frac{2}{n}\sum x_i y_i}{\frac{2}{n}\sum x_i^2} = \frac{\sum x_i y_i}{\sum x_i^2}$
LET $y_i = x_i\beta + \varepsilon_i$
SO
LET $\beta = \frac{\sum x_i y_i}{\sum x_i^2} = \frac{\sum x_i(x_i\beta + \varepsilon_i)}{\sum x_i^2}$
$= \frac{\sum x_i^2\beta + \varepsilon_i x_i}{\sum x_i^2} + \sum e$
$\sum e$ IS THE ERRORS