Functions:

These functions include returning a tweet in lowercase, removing punctuation, removing stop words, and counting the occurance of words.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

stop_words_file = open("stop_words.txt", "r")
words = []
for line in stop_words_file:
  words.append(line.strip())

def clean_data(tweet):
  alphabet = '''abcdefghijklmnopqrstuvwxyz '''
  tweet = str(tweet).lower()
  for i in tweet:
    if i not in alphabet:
      tweet = tweet.replace(i, "")
    return tweet

def remove_stop_words(tweet):
  tweet = tweet.split()
  for i in range(len(tweet)):
    for j in words:
      if tweet[i] == j:
        tweet[i] = ""
  return tweet

def bag_of_words(tweet):
  dic = {}
  for i in tweet:
    if i != "":
      dic[i] = 0
  for i in tweet:
    if i != "":
      dic[i] = dic[i] + 1
  return dic
```

Applying Functions:

```
df = pd.read_csv("us_election_tweets.csv", sep = ',')

df['tweet']
df['tweet'] = df['tweet'].astype(str)
df['tweet'] = df['tweet'].apply(clean_data)
df['tweet'] = df['tweet'].apply(remove_stop_words)
df['tweet'] = df['tweet'].apply(bag_of_words)
df['tweet']
```

```
/usr/local/lib/python3.6/dist-packages/IPython/core/interactiveshell.py:2718:
  interactivity=interactivity, compiler=compiler, result=result)
0         {'americans': 1, 'infuriate': 1, 'comments,': ...
1         {'armstrongcbc': 1, 'surprise.': 1, 'it's': 1,...
2         {'can’t': 1, 'wait': 1, 'till': 1, 'election':...
3         {'america': 1, 'foreign': 2, 'country,': 1, 'c...
4         {'wow.': 1, 'guess': 1, 'actual': 1, 'journali...
                                ...
128148    {'maximebernier': 1, 'setting': 1, 'table': 1,...
128149    {'donald': 1, 'trump': 2, 'seeks': 1, 'fast-tr...
128150    {'daily': 1, 'media,': 1, 'folks:': 1, '#clima...
128151    {'inject': 1, 'rush': 1, 'delivered': 1, 'vacc...
128152    {'expected:': 1, '“trump': 1, 'considers': 1, ...
Name: tweet, Length: 128153, dtype: object
```

Candidate Relation:

```
df_list = df['tweet']

def candidate_relation(tweet):
    candidate = ""
    for page in tweet:
      if "trump" in page:
        if not "T" in candidate:
          candidate = candidate + "T"
      if "biden" in page:
        if not "B" in candidate:
          candidate = candidate + "B"
    if candidate == "":
      candidate = "None"

    return candidate

df['candidate'] = df['tweet'].apply(candidate_relation)
df['candidate']
```

```
0         None
1         None
2            T
3         None
4         None
          ...
128148    None
128149       T
128150    None
128151       T
128152       T
Name: candidate, Length: 128153, dtype: object
```

Candidate Score:

```
corpus = pd.read_csv('corpus.txt', delimiter = "\t", names = ['Word', 'Score'])

corpus_dic = {}
for i in range(len(corpus['Word'])):
  corpus_dic[corpus.loc[i]['Word']] = corpus.loc[i]['Score']

def tweet_score(tweet):
    score = 0
    for word in tweet:
      if word in corpus_dic:
        score = score + corpus_dic[word]
    return score

def normal(score):
  normal = 0
  if score != 0:
    normal = (score - np.min(df['score'])) / (np.max(df['score']) - np.min(df['scor
  else:
    normal = -1
  return normal

df['score'] = df['tweet'].apply(tweet_score)
df['normal'] = df['score'].apply(normal)
```

Analysis:

In order to complete my analysis, I first separated my data into six groups and removed all duplicates, ie. if one user had mulitple tweets related to a single candidate, I found the average of their scores. I did not score neutral scores (where df['normal'] = -1) and Trump|Biden scores because without more sophisticated code, it was impossible to filter if these tweets were positive or negative toward either Trump or Biden. Tweets that did not mention either Trump or Biden were not included into the analysis of the environment beyond a tally to calculate percentages:

```
trump = (df.loc[(df['candidate'] == 'T') & (df['normal'] >= 0)])
new_trump = trump.filter(["username", "normal"], axis = 1)
uniquet = new_trump.groupby('username').mean()

neutral_t = (df.loc[(df['candidate'] == 'T') & (df['normal'] == -1)])
neutral_t = neutral_t.drop_duplicates(subset=["username"])

biden = (df.loc[(df['candidate'] == 'B') & (df['normal'] >= 0)])
new_biden = biden.filter(["username", "normal"], axis = 1)
uniqueb = new_biden.groupby('username').mean()

neutral_b = (df.loc[(df['candidate'] == 'B') & (df['normal'] == -1)])
neutral_b = neutral_b.drop_duplicates(subset=["username"])

trumpbiden = (df.loc[(df['candidate'] == 'TB')])
trumpbiden = trumpbiden.drop_duplicates(subset=["username"])

other = df.loc[((df['candidate'] != 'TB') & (df['candidate'] != 'T') & (df['candida
other = other.drop_duplicates(subset=['username'])
```

I then did a simple percentage calucaltion to understand what percentage of people were Trump positive/negative and what percentage of people were Biden positive/negative. From this code we can see that Trump has considerably more positive tweets than Biden. We can also see that Trump has more negative tweets than Biden has positive tweets.

```
total = len(trumpbiden) + len(other) + len(uniquet) + len(uniqueb) + len(neutral_t)

other_total = len(other) / total

tb_total = len(trumpbiden) / total

uniquet_positive = sum(i > 0.5 for i in uniquet['normal'])
tpositive_total = uniquet_positive / total

uniquet_negative = sum(i < 0.5 for i in uniquet['normal'])
tnegative_total = uniquet_negative / total

uniqueb_positive = sum(i > 0.5 for i in uniqueb['normal'])
bpositive_total = uniqueb_positive / total

uniqueb_negative = sum(i < 0.5 for i in uniqueb['normal'])
bnegative_total = uniqueb_negative / total

neutral_t_total = len(neutral_t) / total
neutral_b_total = len(neutral_b) / total

print("Other %", (other_total * 100))
print("Trump|Biden %: ", (tb_total * 100))
print("Positive Trump %: ", (tpositive_total * 100))
print("Negative Trump %: ", (tnegative_total * 100))
print("Neutral Trump %: ", (neutral_t_total * 100))
print("Positive Biden %: ", (bpositive_total * 100))
print("Negative Biden %: ", (bnegative_total * 100))
print("Neutral Biden %: ", (neutral_b_total * 100))
```

```
Other % 67.74179368550611
Trump|Biden %:  6.321087508379398
Positive Trump %:  7.825532709869557
Negative Trump %:  4.319922634812136
Neutral Trump %:  7.161774563996614
Positive Biden %:  3.40121102893502
Negative Biden %:  0.9846478455333694
Neutral Biden %:  2.1901820939151837
```

Unfortunately, this analysis is misleading as Trump's sample size is 3x that of Biden's. In order to account for this, I took a random sample from Trump's total tweets that is equal to Biden's tweet total. From this, we can more easily see that the twitter environment is slightly more positive toward Biden than Trump. Biden has a higher percentage of positive tweets and a lower percentage of negative tweets than Trump.

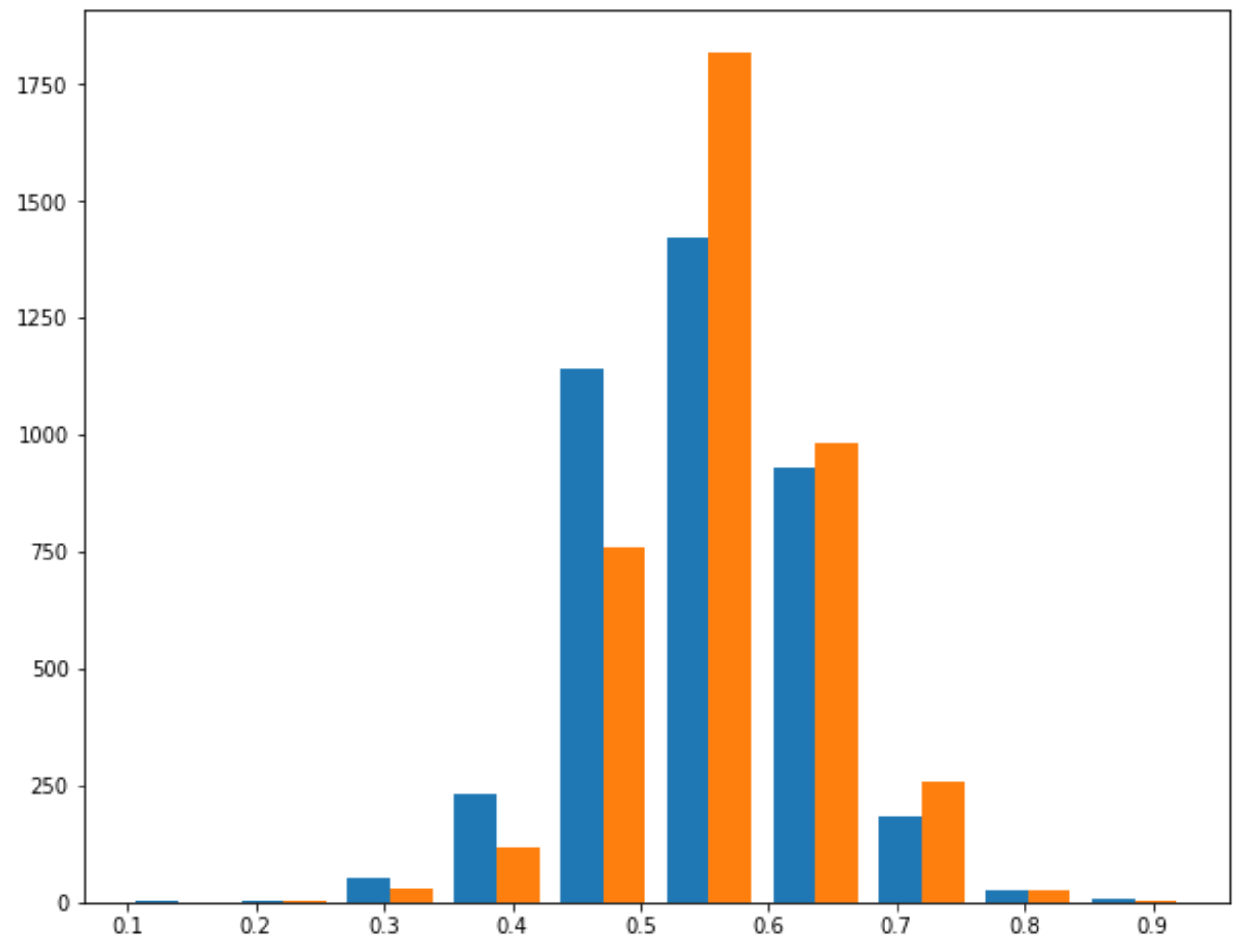
```
sample_uniquet = uniquet.sample(n = 3998)
sample_neutral_t = neutral_t.sample(n = 1993)

total = len(trumpbiden) + len(other) + len(sample_uniquet) + len(uniqueb) + len(sam

uniquet_positive = sum(i > 0.5 for i in sample_uniquet['normal'])
tpositive_total = uniquet_positive / total

uniquet_negative = sum(i < 0.5 for i in sample_uniquet['normal'])
tnegative_total = uniquet_negative / total

uniqueb_positive = sum(i > 0.5 for i in uniqueb['normal'])
bpositive_total = uniqueb_positive / total

uniqueb_negative = sum(i < 0.5 for i in uniqueb['normal'])
bnegative_total = uniqueb_negative / total

neutral_t_total = len(sample_neutral_t) / total
neutral_b_total = len(neutral_b) / total

print("Other %", (other_total * 100))
print("Trump|Biden %: ", (tb_total * 100))
print("Positive Trump %: ", (tpositive_total * 100))
print("Negative Trump %: ", (tnegative_total * 100))
print("Neutral Trump %: ", (neutral_t_total * 100))
print("Positive Biden %: ", (bpositive_total * 100))
print("Negative Biden %: ", (bnegative_total * 100))
print("Neutral Biden %: ", (neutral_b_total * 100))
```

```
Other % 67.74179368550611
Trump|Biden %:  6.321087508379398
Positive Trump %:  3.2477921816143214
Negative Trump %:  1.7725537624248837
Neutral Trump %:  2.510802877407813
Positive Biden %:  3.899114353024176
Negative Biden %:  1.128790455673558
Neutral Biden %:  2.510802877407813
```

Thus I would conclude by saying that the twitter enviornment is more positive toward Biden and more negative toward Trump, but only slightly. This slightness can be seen in a side by side histogram shown below. It more easily shows that Trump's (blue) and Biden's (orange) tweet scores are roughly the same. Biden has a slightly fewer negative scores and slightly more positive scores.

```
plt.figure(figsize=(8,6))
bins = np.linspace(0, 1, 10)

plt.figure(figsize=[10,8])
n, bins, patches = plt.hist([(sample_uniquet['normal']), (uniqueb['normal'])])
```

```
<Figure size 576x432 with 0 Axes>
```



In order to find how many supporters each candidate had, I looked at the total number of positive and negative scores of each candidate. With these numbers, we would assume that Trump has much more support than Biden, however we must also remember that there are many more tweets about Trump than Biden.

There was the possibility of making an assumption that those who negatively view Trump will positively view Biden and vice versa, especially as the United States operates on a two party system and the percentage of those who vote for niether is extremely small. Despite this assumption leading to a non-definitive answer, I produced a test to show an interesting scenario that puts the number of positive sentiment for Trump and for Biden much closer.

```
lentpositive = len(uniquet[(uniquet['normal'] >= 0.5)])
lentnegative = len(uniquet[(uniquet['normal'] <= 0.5)])

lenbpositive = len(uniqueb[(uniqueb['normal'] >= 0.5)])
lenbnegative = len(uniqueb[(uniqueb['normal'] <= 0.5)])

print("Number of Positive Trump Tweets: ", lentpositive)
print("Number of Negative Trump Tweets: ", lentnegative)
print("Number of Positive Biden Tweets: ", lenbpositive)
print("Number of Negative Biden Tweets: ", lenbnegative)
```

```
Number of Positive Trump Tweets:  7163
Number of Negative Trump Tweets:  3973
Number of Positive Biden Tweets:  3102
Number of Negative Biden Tweets:  903
```

```
#if negative tweets for Trump were positive tweets for Biden and vice versa:

print("Number of Positive Trump Tweets: ", lentpositive + lenbnegative)
print("Number of Positive Biden Tweets: ", lentnegative + lenbpositive)
```

```
Number of Positive Trump Tweets:  8066
Number of Positive Biden Tweets:  7075
```

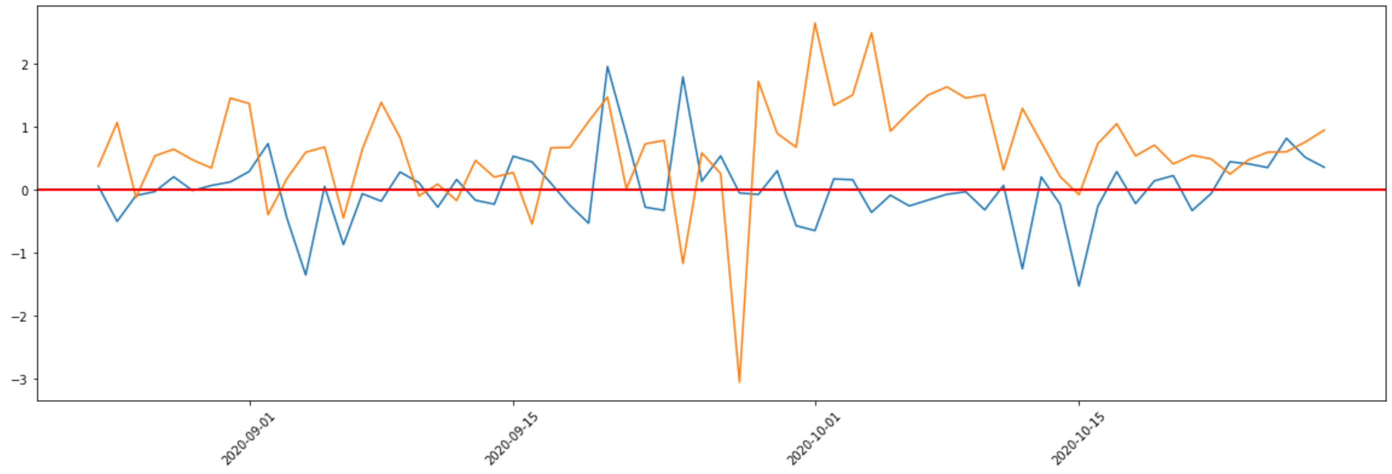
To get a better understanding of the candidate sentiment over the two month period, I plotted the mean score of each candidates based on day. From this, I was able to notice a fairly volatile twitter environment, although staying mainly positive for Biden and hovering on 0 for Trump. There are a couple noticible spikes for each Trump occuring throughout the two month period, three below zero and two above. Through research, we can see that there are news articles about TikToc bans, transition of power, and income tax. Any of these and more could be attributed to the spikes in score.

I also noticed a sharp decline in Biden's score at the end of September followed by a sharp increase at the beginning of October. The beginning of October also happens to be when we see the greatest difference of scores between Trump and Biden, which could be a result of the Presidential Debate that occurs at that time.

```
import matplotlib.dates as mdates

allcandidates = df.groupby(['date', 'candidate'], as_index=False)['score'].mean()
```

```
trumptime = allcandidates.loc[(allcandidates['candidate'] == 'T')]
bidentime = allcandidates.loc[(allcandidates['candidate'] == 'B')]

bidentime["date"] = pd.to_datetime(bidentime["date"])
bidentime = bidentime.sort_values(by="date")
trumptime["date"] = pd.to_datetime(trumptime["date"])
trumptime = trumptime.sort_values(by="date")

plt.figure(figsize=(20,6))
plt.xticks(rotation=45)
plt.plot(trumptime['date'],trumptime['score'])
plt.plot(bidentime['date'], bidentime['score'])
plt.axhline(linewidth=2, color='r')
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  import sys
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:9: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  if __name__ == '__main__':
<matplotlib.lines.Line2D at 0x7ff83187a3c8>
```

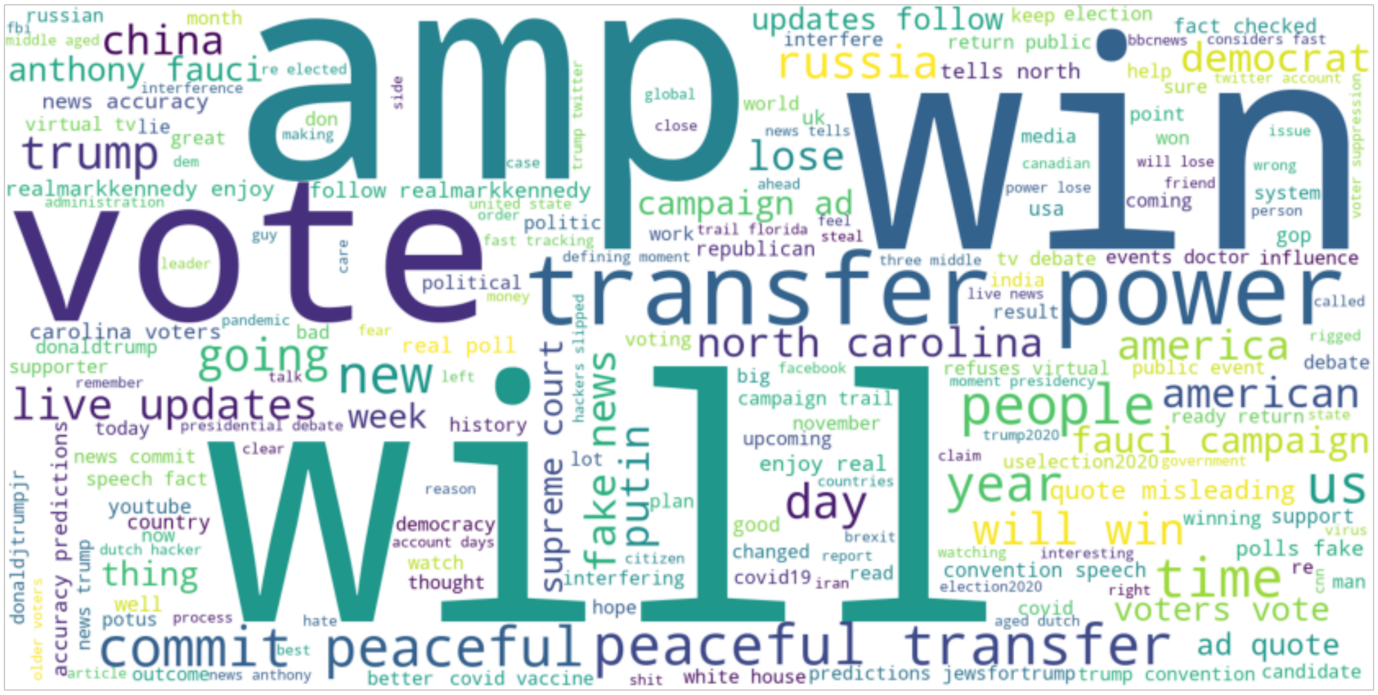


Word Cloud:

Below is the word cloud for tweets related to Trump:

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

stopwords = ["https", "co", "RT", "realdonaldtrump", "trump", "donald", "therealdor
trumpwords = (df.loc[(df['candidate'] == 'T')])

def tweetstring(tweet):
  tweetstring = ""
  for i in tweet:
    # if i not in stopwords:
    tweetstring = tweetstring + " " + i
  return tweetstring

trumpwords['tweetstring'] = trumpwords['tweet'].apply(tweetstring)

trumpwordsstring = np.array(trumpwords['tweetstring'])

tws = ""
for i in trumpwordsstring:
  tws = tws + i

wordcloud = WordCloud(width = 1600, height = 800, background_color ='white', stopwc
plt.figure(figsize=(15,8))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:13: SettingWithCo
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  del sys.path[0]
```

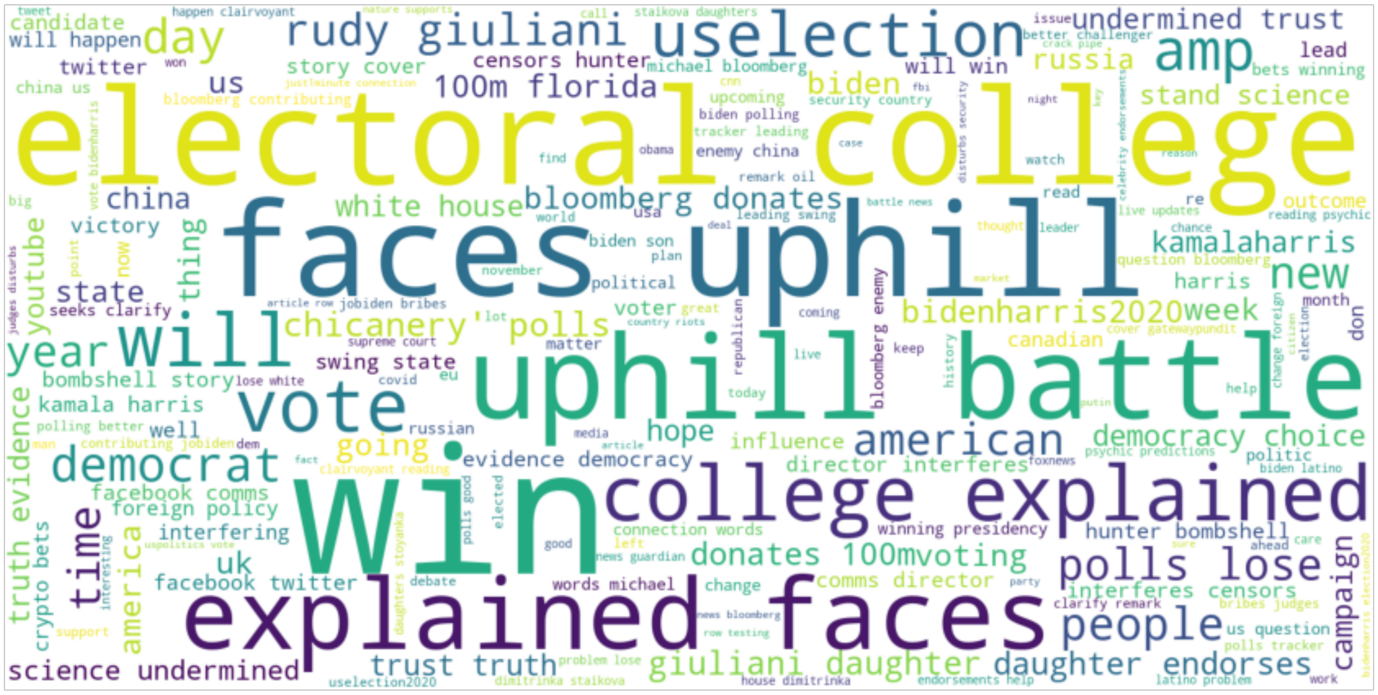


Below is the word cloud for tweets related to Biden:

```
stopwords = ["https", "co", "RT", "biden", "joe", "joebiden", "bbc", "president", '
bidenwords = (df.loc[(df['candidate'] == 'B')])

def tweetstring(tweet):
  tweetstring = ""
  for i in tweet:
    # if i not in stopwords:
    tweetstring = tweetstring + " " + i
  return tweetstring

bidenwords['tweetstring'] = bidenwords['tweet'].apply(tweetstring)

bidenwordsstring = np.array(bidenwords['tweetstring'])

bws = ""
```

```
for i in bidenwordsstring:
  bws = bws + i

wordcloud = WordCloud(width = 1600, height = 800, background_color ='white', stopwo
plt.figure(figsize=(15,8))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:11: SettingWithCo
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  # This is added back by InteractiveShellApp.init_path()
```


candidate
will happen
day
rudy giuliani
uselection
undermined trust
amp
lead
russia
bets winning
twitter
us
story cover
censors hunter
michael bloomberg
biden
will win
china us
bloomberg contributing
100m florida
upcoming
stand science
electoral college
china
white house
bloomberg donates
outcome
victory
faces uphill
kamalaharris
youtube
thing
new
state
harris
seeks clarify
year
will
chicanery
polls
voter
bidenharris2020week
month
canadian
bombshell story
vote
uphill battle
kamala harris
swing state
well
russian
hope
american
democracy choice
democrat
going
evidence democracy
influence
director interferes
facebook comms
foreign policy
win
college explained
truth evidence
time
interfering
winning presidency
hunter bombshell
crypto bets
uk
donates 100mvoting
polls lose
america
facebook twitter
comms director
interferes censors
explained faces
people
campaign
science undermined
trust truth
giuliani daughter
daughter endorses