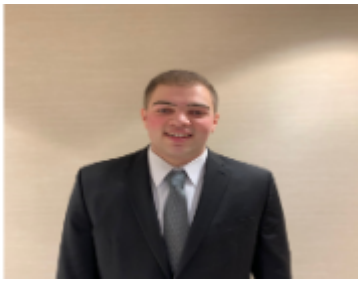
# Table of Contents

# Final Project Phase 4

## Team Information


- Group 2: Pied Piper
- President Trump Twitter Analysis & Election Simulation
- Alex Bzdel - abzdel@bryant.edu
- Zach Galante - zgalante@bryant.edu
- Robert Mitchell - rmitchell2@bryant.edu

# Abstract

We plan to explore Donald Trump’s approval rating based on tweets regarding the US stimulus checks as well as tweets regarding coronavirus. We will explore how Trump's approval rating moves across time in conjunction with these tweets. Our goal is to see if there is a correlation between the number (and sentiment) of tweets with a keyword and the approval rating. We will annotate key dates from the Trump Administration's coronavirus timeline on our plots as well. We will then take the results from this analysis and see how they correlate to the winner of our election simulation.

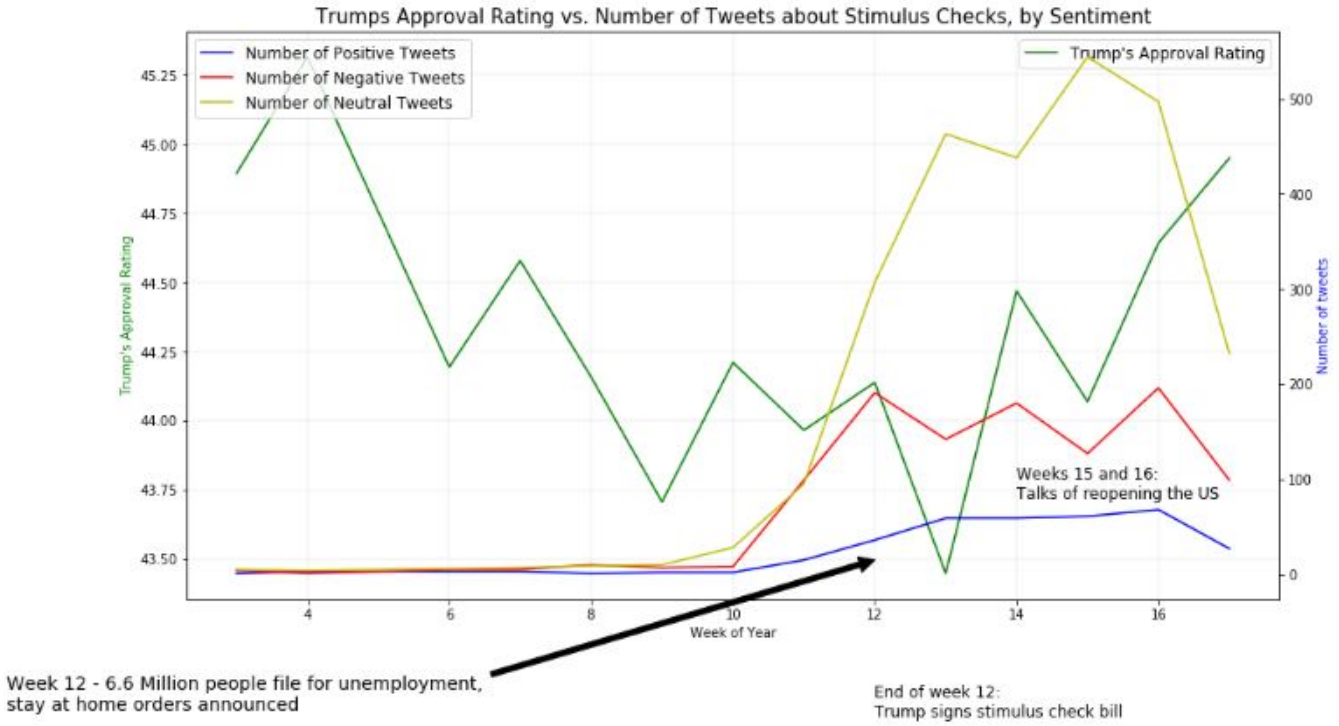
Part 1 will deal with Trump's approval rating as compared to our scraped tweets
Part 2 will produce an election simulation based off of current tweets (sample of 10 states)

## Keywords & hashtags:

- keywords: Trump, Stimulus Check, $1200, covid19, coronavirus
- hashtags: #trump, #stimuluscheck, #money

## Hypotheses

1) President Trump’s approval rating has increased with the number of “positive sentiment” tweets about the stimulus checks distributed by the US government. The more tweets involving stimulus checks that have highly rated words as determined by sentiment analysis, the higher we believe Trump’s approval rating will be.
2) Sentiment ratings for the tweets will show an increase after the stimulus checks actually start to be distributed.
3) That Trump’s approval rating is going to reflect the current state of the country and that it will also be reflected in the results of the 2020 election.




One of our final plots on Trump's Approval Rating vs Stimulus Check Tweets is pictured above

```
Trump Wins Pennsylvania
Trump Vote Count: 81
--------------------------------------------------
Biden Wins Illinois
Biden Vote Count: 98
--------------------------------------------------
Trump Wins Ohio
Trump Vote Count: 97
--------------------------------------------------
Biden Wins Michigan
Biden Vote Count: 112
--------------------------------------------------
Trump Wins Georgia
Trump Vote Count: 111
--------------------------------------------------
Biden Wins North Carolina
Biden Vote Count: 125
--------------------------------------------------
  *****Biden wins the 2020 election*****
--------------------------------------------------
Trump's final score was 111
Biden's final score was 125
--------------------------------------------------
```

Final simulation example above

# Part 1: Tracking Trump's Approval Rating

In [146]:
```
# imports
import numpy as np
import pandas as pd
#import tweepy
#import twython

import json
import csv
import os
import codecs
import time
import gender_guesser.detector as gender
from textblob import TextBlob
from wordcloud import WordCloud, STOPWORDS
import seaborn as sns
import matplotlib.pyplot as plt
import re
%matplotlib inline
```

# Description of the Data - Part 1

## Scraper(s) used

In [147]:
```
#--------- NOTE THAT THIS CELL IS NOT SUPPOSED TO RUN ------------------
import time
from random import seed
from random import randint
# seed random number generator
seed(1)
from datetime import date, datetime, timedelta

def datespan(startDate, endDate, delta=timedelta(days=7)):
    currentDate = startDate
    while currentDate < endDate:
        yield currentDate
        currentDate += delta

text_query = "Stimulus Check"
start_of_period = date(2020, 1, 4)
end_of_period   = date(2020, 5, 2)
start_of_week = start_of_period
week_plus_1_start = start_of_period + timedelta(days=7)

for i, week_plus_1_start in enumerate(datespan(week_plus_1_start, end_of_perio
d, delta=timedelta(days=7))):
    print(f"week {i}, start of week: {start_of_week} {week_plus_1_start}")
    get_tweet_result(text_query = text_query, since_date = f"{start_of_week}",
                until_date = f"{week_plus_1_start}", count = 500)
    time.sleep(randint(20, 90))

    start_of_week = week_plus_1_start
```

```
week 0, start of week: 2020-01-04 2020-01-11

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-147-6a8d3fb3b268> in <module>
     21 for i, week_plus_1_start in enumerate(datespan(week_plus_1_start, end
_of_period, delta=timedelta(days=7))):
     22     print(f"week {i}, start of week: {start_of_week} {week_plus_1_sta
rt}")
---> 23     get_tweet_result(text_query = text_query, since_date = f"{start_o
f_week}",
     24                 until_date = f"{week_plus_1_start}", count = 500)
     25     time.sleep(randint(20, 90))

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

## Attempt at pulling user data (failed scraper)

Citation for scraper: https://github.com/taspinar/twitterscraper (https://github.com/taspinar/twitterscraper)

In [148]:

```
#--------- NOTE THAT THIS CELL IS NOT SUPPOSED TO RUN ------------------
from twitterscraper.query import query_user_info
import pandas as pd
from multiprocessing import Pool
import time
from IPython.display import display
import random


global twitter_user_info
twitter_user_info=[]


def get_user_info(twitter_user):
    """
    An example of using the query_user_info method
    :param twitter_user: the twitter user to capture user data
    :return: twitter_user_data: returns a dictionary of twitter user data
    """
    user_info = query_user_info(user= twitter_user)
    twitter_user_data = {}
    twitter_user_data["user"] = user_info.user
    twitter_user_data["fullname"] = user_info.full_name
    twitter_user_data["location"] = user_info.location
    twitter_user_data["blog"] = user_info.blog
    twitter_user_data["date_joined"] = user_info.date_joined
    twitter_user_data["id"] = user_info.id
    twitter_user_data["num_tweets"] = user_info.tweets
    twitter_user_data["following"] = user_info.following
    twitter_user_data["followers"] = user_info.followers
    twitter_user_data["likes"] = user_info.likes
    twitter_user_data["lists"] = user_info.lists

    return twitter_user_data


def main():
    start = time.time()
   # users = list(df.username.values) # pass list of usernames into main() fun
ction
    #users = ['A3Patriot']
    users = ['A3Patriot', 'whaley1212', 'Janb723Branam', 'jonsaxon67',
       'JLaroc318', 'MamaR130', 'cjstockton67', 'NotTheMacAnon1',
       'VBrown13245591', 'bradyswenson', 'NoelJTom', 'glasscock_keny',
       'cajunvincent', 'grumpy_idiot', 'fwdcrocblu', 'TerryAnastasio',
       'LMenssen', 'AmericanEarlR', 'Frankd06830', 'cchriss147',
       'mtrwf11', 'Isaac_Visage', None]
    pool = Pool(8)
    for user in pool.map(get_user_info,users):
        if True: # if/else meant to fix the following error: NoneType object h
as no attribute 'User'
            twitter_user_info.append(user)
            time.sleep(random.randint(1,10))
        elif False:
            print('an error has occurred')
            break
```

```
        # while true else break

    cols=['id','fullname','date_joined','location','blog', 'num_tweets','follo
wing','followers','likes','lists']
        #cols=['id','date_joined','location','num_tweets','followers']

data_frame = pd.DataFrame(twitter_user_info, index=main().users, columns=cols)
        # save data_frame as a global variable after running this
data_frame.index.name = "Users"
data_frame.sort_values(by="followers", ascending=False, inplace=True, kind='qu
icksort', na_position='last')
elapsed = time.time() - start
print(f"Elapsed time: {elapsed}")
display(data_frame)


if __name__ == '__main__':
    main()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-148-72ca092c163d> in <module>
      1 #--------- NOTE THAT THIS CELL IS NOT SUPPOSED TO RUN ---------------
----
----> 2 from twitterscraper.query import query_user_info
      3 import pandas as pd
      4 from multiprocessing import Pool
      5 import time

ModuleNotFoundError: No module named 'twitterscraper'
```

## Data Description

Our first analysis involves data with tweets involving keywords dating back to January 2020

```
* meant to analyze tweet counts and sentiments across time
* drawback: no user data available (followers, location, etc.)
```

- **We scraped the following tweet characteristics via GenOldTweets**:
  - Formatted Date of tweet
  - Author ID (user ID)
  - Username
  - User Location
  - Tweet Text
  - Hashtags
  - Mentions
  - to
    - who the tweet is addressed to - if anyone
  - Tweet URL
  - Replies
  - Retweets
  - Favorites
  - Permalink
- **We then added the following**:
  - filtered_text
  - retweet_flags
  - Polarity_Score
  - SubjectivityScore
  - sentimentLabel

  #### Third-Party Data For our third party data, we will be using Donald Trump's approval rating across the aforementioned timespan (Jan-Present). We will primarily be using this to compare the first dataset to his approval rating and see how they compare. This data is from fivethirtyeight.com

**Data Processing Tasks**

For both sets of data, a big step in processing is sentiment analysis. We will be producing visualizations of the number of tweets about different keywords in the context of sentiment. Additionally, we will be binning every week (there are 17) for our visualization x-axes to depict how sentiments (of both certain keywords and total tweets) and approval ratings have changed across time. We will also be creating three sub-DataFrames (one for each sentiment) and search for keywords again to get a more detailed view on the sentiment of tweets vs. Trump's Approval Rating

**When and How Long You Scraped Twitter**

- we scraped twitter on 4/30 and 5/1 for an hour each day split between each of our three group members

## Load in Data

In [149]:

```
import pandas as pd
import numpy as np
hashtagTrump = pd.read_csv("Group_2_Phase_4_#Trump_0k_tweets.csv")
hashtagTrump.columns = ['formatted_date', "author_id",  "username",  "geo", "t
ext", "hashtags", 'mentions', "to", "urls",  'replies', 'retweets',  "favorite
s", 'permalink']
new_money = pd.read_csv("Group_2_Phase_4_$1200_0k_tweets.csv")
new_money.columns = ['formatted_date', "author_id",  "username",  "geo", "tex
t", "hashtags", 'mentions', "to", "urls",  'replies', 'retweets',  "favorites"
, 'permalink']
a = new_money.append(hashtagTrump, ignore_index = True)
hash_stimCheck = pd.read_csv("Group_2_Phase_4_Copy of #stimuluscheck_0k_tweet
s.csv")
hash_stimCheck.columns = ['formatted_date', "author_id",  "username",  "geo",
"text", "hashtags", 'mentions', "to", "urls",  'replies', 'retweets',  "favori
tes", 'permalink']
b = a.append(hash_stimCheck, ignore_index = True)
coronavirus = pd.read_csv("Group_2_Phase_4_coronavirus_0k_tweets.csv")
coronavirus.columns = ['formatted_date', "author_id",  "username",  "geo", "te
xt", "hashtags", 'mentions', "to", "urls",  'replies', 'retweets',  "favorite
s", 'permalink']
c = b.append(coronavirus, ignore_index = True)
COVID19 = pd.read_csv("Group_2_Phase_4_COVID19_0k_tweets.csv")
COVID19.columns = ['formatted_date', "author_id",  "username",  "geo", "text",
"hashtags", 'mentions', "to", "urls",  'replies', 'retweets',  "favorites", 'p
ermalink']
d = c.append(COVID19, ignore_index = True)
stim_check = pd.read_csv("Group_2_Phase_4_Stimulus_Check_0k_tweets.csv")
stim_check.columns = ['formatted_date', "author_id",  "username",  "geo", "tex
t", "hashtags", 'mentions', "to", "urls",  'replies', 'retweets',  "favorites"
, 'permalink']
e = d.append(stim_check, ignore_index = True )
Trump = pd.read_csv("Group_2_Phase_4_Trump_0k_tweets.csv")
Trump.columns= ['formatted_date', "author_id",  "username",  "geo", "text", "h
ashtags", 'mentions', "to", "urls",  'replies', 'retweets',  "favorites", 'per
malink']
df = e.append(Trump, ignore_index = True)
df.head()
```

Out[149]:

| | formatted_date | author_id | username | geo | text | hashtags | mentions | |
|---|---|---|---|---|---|---|---|---|
| **0** | Fri Apr 10 23:59:57 +0000 2020 | 2.716171e+08 | ForeverMe_MsB | NaN | I got my 1200$y'all | NaN | NaN | N |
| **1** | Fri Apr 10 23:59:48 +0000 2020 | 4.970938e+07 | _carceeexoxo | NaN | Idk why ppl who probably have 5.00 in their ac... | NaN | NaN | N |
| **2** | Fri Apr 10 23:59:45 +0000 2020 | 1.239146e+08 | bitcoinization | NaN | The open market can sell their Bitcoins as muc... | NaN | NaN | mikealf |
| **3** | Fri Apr 10 23:59:44 +0000 2020 | 2.826924e+09 | 4rdaSquad | NaN | I'll slide all 1200 wassup | NaN | NaN | TeannaTru |
| **4** | Fri Apr 10 23:59:39 +0000 2020 | 5.381119e+08 | Felipe__ | NaN | That $1200 just hit for me check dm | NaN | NaN | TeannaTru |

### Approval rating data

In [150]:
```
approval_df = pd.read_csv('Group_2_Phase_4_pollist.csv')
approval_df.head()
```

Out[150]:

| | president | subgroup | modeldate | startdate | enddate | pollster | grade | samplesize | population |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Donald Trump | Voters | 4/30/2020 | 5/28/2019 | 4/30/2020 | YouGov | B- | 796 | rv |
| **1** | Donald Trump | Voters | 4/30/2020 | 5/29/2019 | 4/30/2020 | YouGov | B- | 771 | rv |
| **2** | Donald Trump | Voters | 4/30/2020 | 5/30/2019 | 4/29/2020 | YouGov | B- | 734 | rv |
| **3** | Donald Trump | Voters | 4/30/2020 | 5/31/2019 | 4/29/2020 | YouGov | B- | 740 | rv |
| **4** | Donald Trump | Voters | 4/30/2020 | 6/1/2019 | 4/29/2020 | YouGov | B- | 728 | rv |

5 rows × 22 columns

# Data Cleaning

In [151]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.6/site-packages
(3.4.5)
Requirement already satisfied: six in /usr/local/lib/python3.6/site-packages
(from nltk) (1.14.0)
```

In [152]:
```
# deal with nots
!pip install TextBlob
```

```
Requirement already satisfied: TextBlob in /usr/local/lib/python3.6/site-pack
ages (0.15.3)
Requirement already satisfied: nltk>=3.1 in /usr/local/lib/python3.6/site-pac
kages (from TextBlob) (3.4.5)
Requirement already satisfied: six in /usr/local/lib/python3.6/site-packages
(from nltk>=3.1->TextBlob) (1.14.0)
```

In [153]:
```
import numpy as np
import pandas as pd
#import tweepy
#import twython

import json
import csv
import os
import codecs
import time
import gender_guesser.detector as gender
from textblob import TextBlob
from wordcloud import WordCloud, STOPWORDS
import seaborn as sns
import matplotlib.pyplot as plt
import re
%matplotlib inline
```

In [154]:
```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[154]:
```
True
```

In [155]:
```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re
stop_words = set(stopwords.words('english'))


def remove_url(txt):
    """Replace URLs found in a text string with nothing
    (i.e. it will remove the URL from the string).
    """
    return re.sub("([^0-9A-Za-z \t])|(\w+:\/\/\S+)", "", txt)

def remove_stop_words(txt):
    """lower case the text, and DROP stop words. """
    return " ".join([w for w in word_tokenize(txt.lower()) if not w in stop_wo
rds])

def preprocess_tweet_text(txt):
    return remove_stop_words(remove_url(txt))
```

In [156]:
```
df.text.isnull().sum()
df.text = df.text.fillna(' ') # fill nulls w/ empty string
```

In [157]:
```
# Remove URLs and Stop Words
df['filtered_text'] = df.text.apply(preprocess_tweet_text)

#retweet
df['retweet_flags'] = df.text.str.startswith('RT')
```

## Create week bins

In [158]:
```
approval_df.enddate = pd.to_datetime(approval_df.enddate)
approval_df['week_bins'] = approval_df.enddate.dt.week
```

In [159]:
```
approval_df = approval_df.where(approval_df['week_bins']!=18)
```

In [160]:
```
df.formatted_date = pd.to_datetime(df.formatted_date)
df['week_bins'] = df.formatted_date.dt.week
```

## Calculate Polarity

In [161]:
```
df['PolarityScore'] = df.filtered_text.apply(lambda txt:  TextBlob(txt).polari
ty)
df['SubjectivityScore'] = df.filtered_text.apply(lambda txt:  TextBlob(txt).su
bjectivity)
```

In [162]:
```
def sentiment_bins(data):
    if data <= -0.5:
        grouping = 'Strong-Negative'
    elif data > -0.5 and data < 0.0:
        grouping = 'Mild-Negative'
    elif data > 0.0 and data < 0.5:
        grouping = 'Mild Positive'
    elif data >=0.5:
        grouping = 'Strong-Positive'
    else:
        grouping = 'Neutral'
    return grouping

df['sentimentLabel'] = df['PolarityScore'].apply(sentiment_bins)
#These bins should be  used later for graphing results.
```

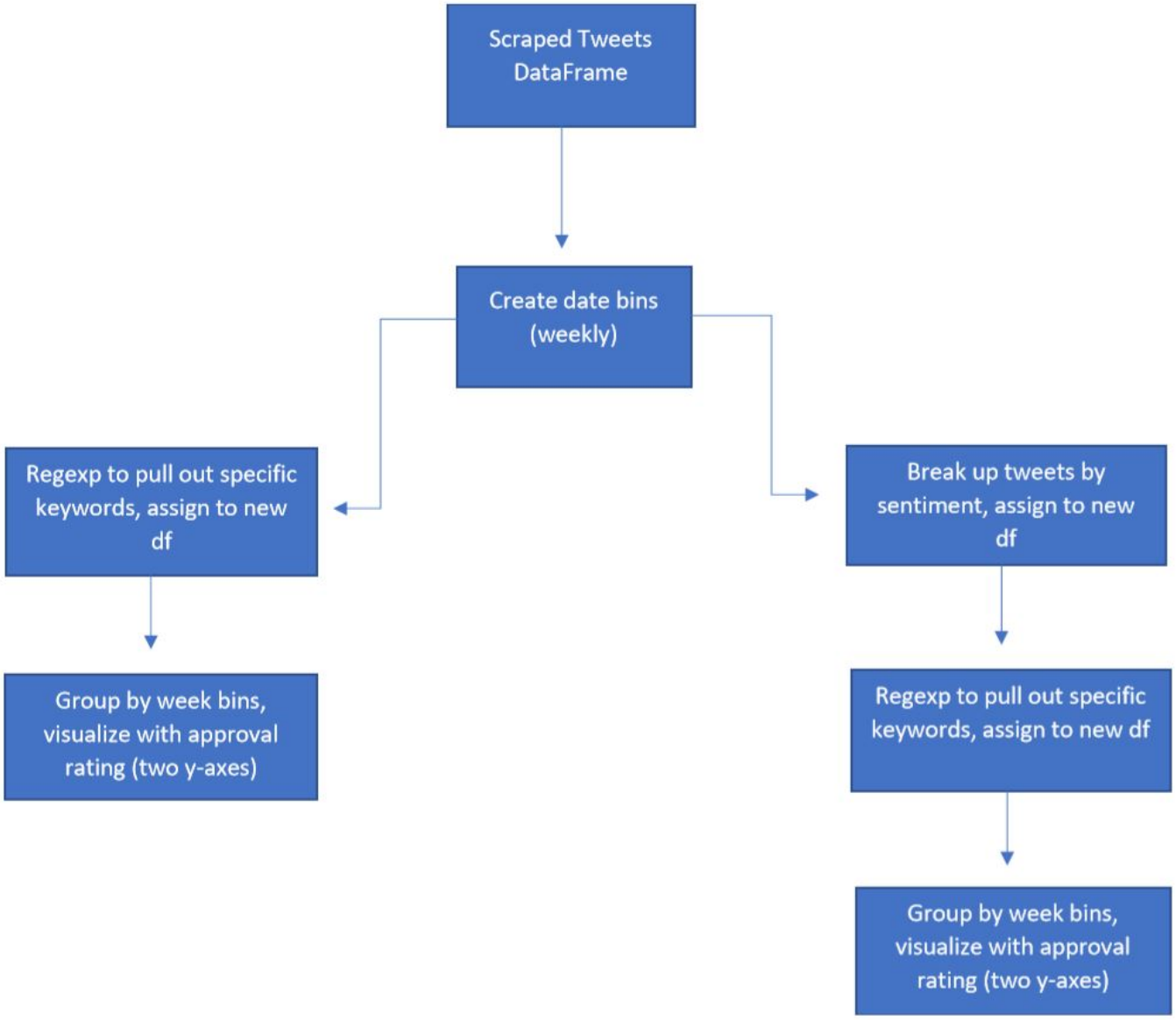
In [163]:
```
df['sentiment'] = df.PolarityScore.apply(sentiment_bins)
df.head()
```

Out[163]:

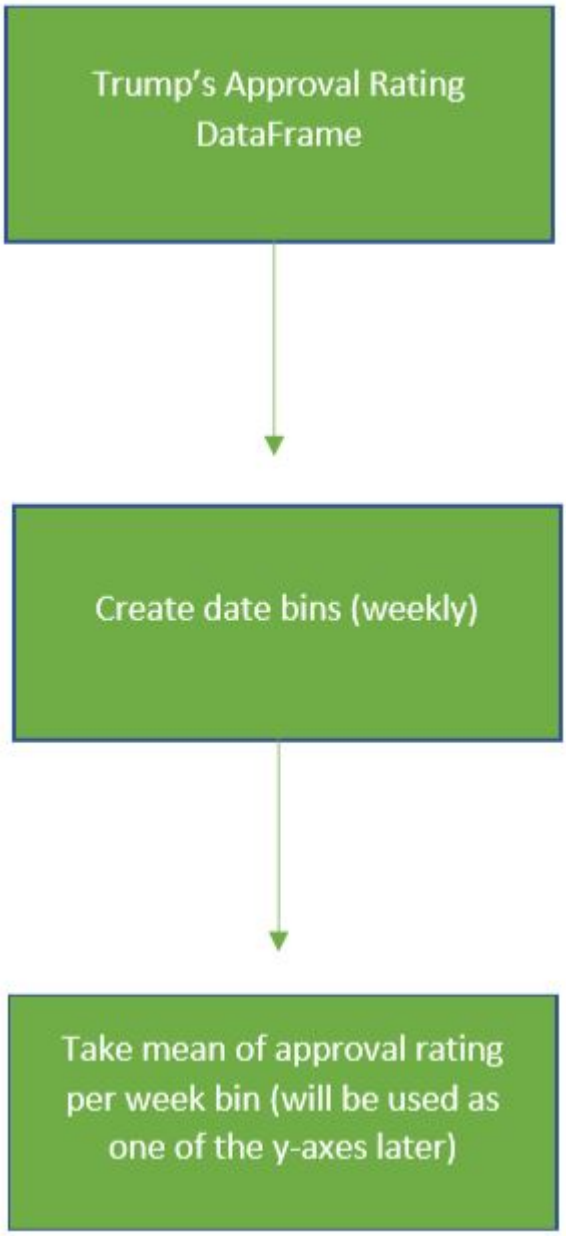
| | formatted_date | author_id | username | geo | text | hashtags | mentions | |
|---|---|---|---|---|---|---|---|---|
| **0** | 2020-04-10 23:59:57+00:00 | 2.716171e+08 | ForeverMe_MsB | NaN | I got my 1200$y'all | NaN | NaN | N |
| **1** | 2020-04-10 23:59:48+00:00 | 4.970938e+07 | _carceeexoxo | NaN | Idk why ppl who probably have 5.00 in their ac... | NaN | NaN | N |
| **2** | 2020-04-10 23:59:45+00:00 | 1.239146e+08 | bitcoinization | NaN | The open market can sell their Bitcoins as muc... | NaN | NaN | mikealf |
| **3** | 2020-04-10 23:59:44+00:00 | 2.826924e+09 | 4rdaSquad | NaN | I'll slide all 1200 wassup | NaN | NaN | TeannaTru |
| **4** | 2020-04-10 23:59:39+00:00 | 5.381119e+08 | Felipe__ | NaN | That $1200 just hit for me check dm | NaN | NaN | TeannaTru |

# Workflow Diagrams

First, let's look at our planned workflow for how we will use our scraped tweets DataFrame




Here's how we plan to use our third party data about Trump's approval rating




## Visual EDA Prep - Part 1

## Stimulus check tweets (regexp)

In [164]:
```
import re
df.text = df.text.fillna(" ")
stimuluscheck_df = df[df['text'].str.contains("stimulus check|#stimuluscheck",
flags=re.IGNORECASE)]
```

In [165]:
```
stimuluscheck_df.groupby('week_bins')['text'].count().values #example of the c
ode we will use to plot this
```

Out[165]:
```
array([   2,    5,   16,   16,   18,   24,   21,   31,   33,   59,  293,
        747,  980,  963, 1030, 1031,  477])
```

## Coronavirus tweets

In [166]:
```
corona_df = df[df['text'].str.contains("1200", flags=re.IGNORECASE)]
arrcorona = corona_df.groupby('week_bins')['text'].count().values

arrcorona= np.insert(arrcorona, 0, 0) #insert 0 at index 0 b/c missing data
```

## Count of tweets for each sentiment

In [167]:
```
df_pos = df.where((df['sentimentLabel']=='MildPositive') | (df['sentimentLabe
l']==('Positive')) | (df['sentimentLabel']==('Strong-Positive')))
df_neg = df.where((df['sentimentLabel']=='Mild-Negative') | (df['sentimentLabe
l']==('Negative')) | (df['sentimentLabel']==('Strong-Negative')))
df_neut = df.where((df['sentimentLabel']=='Neutral'))
```

In [168]:
```
df_pos = df_pos.dropna(subset=['sentimentLabel'])
df_neg = df_neg.dropna(subset=['sentimentLabel'])
df_neut = df_neut.dropna(subset=['sentimentLabel'])
```

In [169]:
```
df_pos.groupby('week_bins')['sentimentLabel'].count()
df_neg.groupby('week_bins')['sentimentLabel'].count()
df_neut.groupby('week_bins')['sentimentLabel'].count()
```

Out[169]:
```
week_bins
1.0        2
2.0     1101
3.0     1108
4.0     1179
5.0     1164
6.0     1200
7.0     1137
8.0     1138
9.0     1372
10.0    1333
11.0    1967
12.0    2002
13.0    1984
14.0    2069
15.0    2115
16.0    2022
17.0    1597
Name: sentimentLabel, dtype: int64
```

In [170]:
```
arrneg = df_neg.groupby('week_bins')['text'].count().values
arrneg = np.insert(arrneg, 0, 0)

arrpos = df_pos.groupby('week_bins')['text'].count().values

arrneut = df_neut.groupby('week_bins')['text'].count().values
```

## Stimulus check by sentiment

In [171]:
```
stimuluscheck_pos_df = df_pos[df_pos['text'].str.contains("stimulus check|#sti
muluscheck", flags=re.IGNORECASE)]
stimuluscheck_neg_df = df_neg[df_neg['text'].str.contains("stimulus check|#sti
muluscheck", flags=re.IGNORECASE)]
stimuluscheck_neut_df = df_neut[df_neut['text'].str.contains("stimulus check|#
stimuluscheck", flags=re.IGNORECASE)]
```

Tweets with positive sentiment are missing week bins 1, 2, and 5, so we'll temporarily drop them for all sentiment queries

In [172]:
```
approval_temp_y = approval_df.groupby('week_bins')['approve'].mean()
approval_temp_y = approval_temp_y.drop(labels=[1,2,5])

approval_temp_x = approval_df.groupby('week_bins')['approve'].mean().index
approval_temp_x = approval_temp_x.drop(labels=[1,2,5])

stimcheckneg = stimuluscheck_neg_df.groupby('week_bins')['sentimentLabel'].cou
nt()
stimcheckneg = stimcheckneg.drop(labels=[5])

stimcheckneut = stimuluscheck_neut_df.groupby('week_bins')['sentimentLabel'].c
ount()
stimcheckneut = stimcheckneut.drop(labels=[1,2,5])
```

## Coronavirus tweets by sentiment

In [173]:
```
# search for coronavirus tweets by sentiment

corona_pos_df = df_pos[df_pos['text'].str.contains("coronavirus|covid19", flag
s=re.IGNORECASE)]
corona_neg_df = df_neg[df_neg['text'].str.contains("coronavirus|covid19", flag
s=re.IGNORECASE)]
corona_neut_df = df_neut[df_neut['text'].str.contains("coronavirus|covid19", f
lags=re.IGNORECASE)]
```

In [174]:
```
corona_pos_df.groupby('week_bins')['sentimentLabel'].count()
corona_neg_df.groupby('week_bins')['sentimentLabel'].count()
corona_neut_df.groupby('week_bins')['sentimentLabel'].count()
```

Out[174]:
```
week_bins
2.0     169
3.0     253
4.0     323
5.0     274
6.0     283
7.0     301
8.0     310
9.0     592
10.0    526
11.0    650
12.0    694
13.0    708
14.0    719
15.0    668
16.0    687
17.0    541
Name: sentimentLabel, dtype: int64
```

In [175]:
```
corona_temp_y = approval_df.groupby('week_bins')['approve'].mean()
corona_temp_y = corona_temp_y.drop(labels=[1])

corona_temp_x = approval_df.groupby('week_bins')['approve'].mean().index
corona_temp_x = corona_temp_x.drop(labels=[1])
```

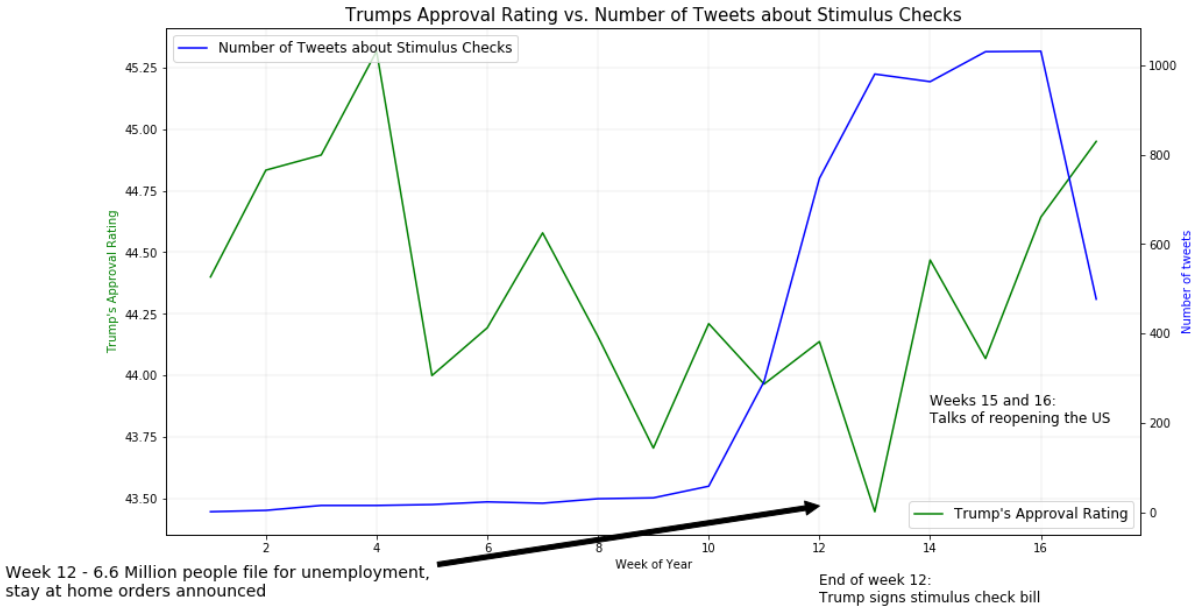
# Visuals and EDA - Part 1

## Number of Stimulus Check Tweets

First, let's explore how Trump's approval rating has changed with the number of tweets about stimulus checks since the beginning of the year

In [176]:

```
import numpy as np
import matplotlib.pyplot as plt
x = approval_df.groupby('week_bins')['approve'].mean().index
y1 = approval_df.groupby('week_bins')['approve'].mean()
y2 = stimuluscheck_df.groupby('week_bins')['text'].count().values
#y3 = arr1200

fig, ax1 = plt.subplots(figsize=(15,8))

ax1.set_title('Trumps Approval Rating vs. Number of Tweets about Stimulus Chec
ks', fontsize=15)
ax1.grid(color='grey', linestyle='-', linewidth=0.25, alpha=0.5)

#duplicate ax1
ax2 = ax1.twinx()

#ax1 will plot Trump's approval rating, ax2 will plot number of tweets
ax1.plot(x, y1, 'g-')
ax2.plot(x, y2, 'b-')
#ax2.plot(x, y3, 'r-')

# set legends
ax1.legend(["Trump's Approval Rating"], fontsize=12,loc=4)
ax2.legend(["Number of Tweets about Stimulus Checks"], fontsize=12, loc=2)
#ax2.legend(bbox_to_anchor=(0.04, 0.82, 1., .102),labelspacing=0.1,
            #handlelength=0.1, handletextpad=0.1,frameon=False, ncol=4, column
spacing=0.7)


ax1.set_xlabel('Week of Year')
ax1.set_ylabel("Trump's Approval Rating", color='g')
ax2.set_ylabel("Number of tweets", color='b')

# plt.text(6.4,3.3,"""Note on our visualization: We made it so that when there
is a
#          null value we replace it with the rank 15 so if an artist is not
#          on that platform's top 10 they go to the bottom""", fontsize = 10)
plt.text(14,200, """Weeks 15 and 16:
Talks of reopening the US""", fontsize = 12)

plt.annotate("""Week 12 - 6.6 Million people file for unemployment,
stay at home orders announced""",
             xy = (12,15), xytext = (2,15), textcoords = ('figure points'),
             arrowprops=dict(facecolor='black'), fontsize = 14)

plt.text(12,-200, """End of week 12:
Trump signs stimulus check bill""", fontsize = 12)




plt.show()

# print('notable dates:')
# print('Week 12 - 6.6 Million people file for unemployment')
```

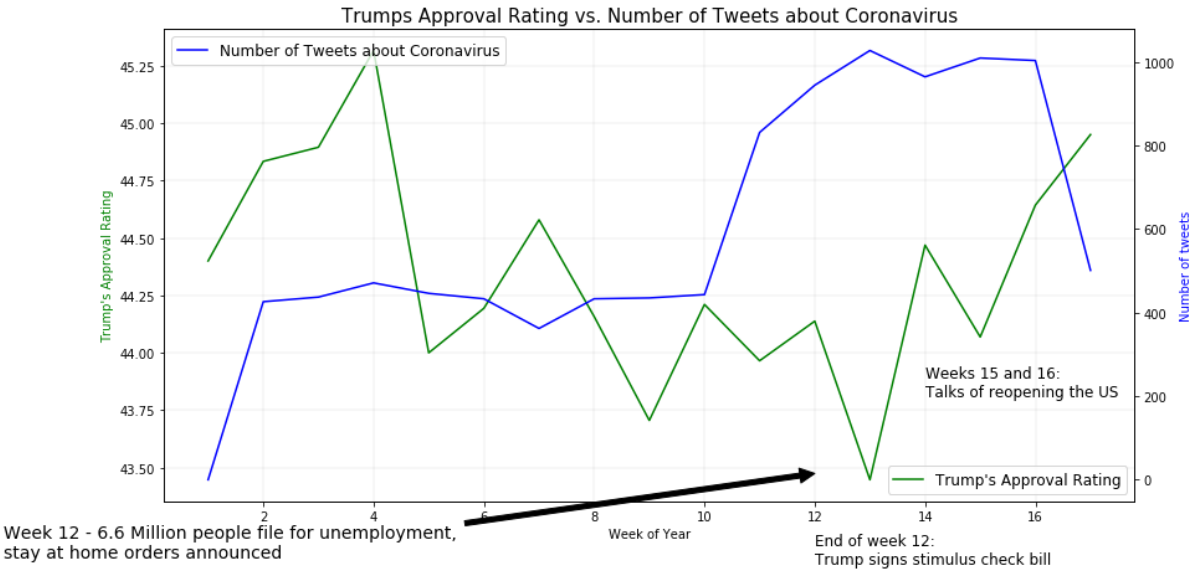



## Number of Coronavirus Tweets

Now, let's look into how his rating changes with number of tweets about coronavirus

In [177]:

```
import numpy as np
import matplotlib.pyplot as plt
x = approval_df.groupby('week_bins')['approve'].mean().index
y1 = approval_df.groupby('week_bins')['approve'].mean()
y2 = arrcorona

fig, ax1 = plt.subplots(figsize=(14,7))

ax1.set_title('Trumps Approval Rating vs. Number of Tweets about Coronavirus',
fontsize=15)
ax1.grid(color='grey', linestyle='-', linewidth=0.25, alpha=0.5)

#duplicate ax1
ax2 = ax1.twinx()

#ax1 will plot Trump's approval rating, ax2 will plot number of tweets
ax1.plot(x, y1, 'g-')
ax2.plot(x, y2, 'b-')

# set legends
ax1.legend(["Trump's Approval Rating"], fontsize=12, loc=4)
ax2.legend(["Number of Tweets about Coronavirus"], fontsize=12, loc=2)


plt.text(14,200, """Weeks 15 and 16:
Talks of reopening the US""", fontsize = 12)

plt.annotate("""Week 12 - 6.6 Million people file for unemployment,
stay at home orders announced""",
             xy = (12,15), xytext = (2,15), textcoords = ('figure points'),
             arrowprops=dict(facecolor='black'), fontsize = 14)

plt.text(12,-200, """End of week 12:
Trump signs stimulus check bill""", fontsize = 12)



ax1.set_xlabel('Week of Year')
ax1.set_ylabel("Trump's Approval Rating", color='g')
ax2.set_ylabel("Number of tweets", color='b')
#ax.legend((y1, y2), ('label1', 'label2'))
plt.show()
```

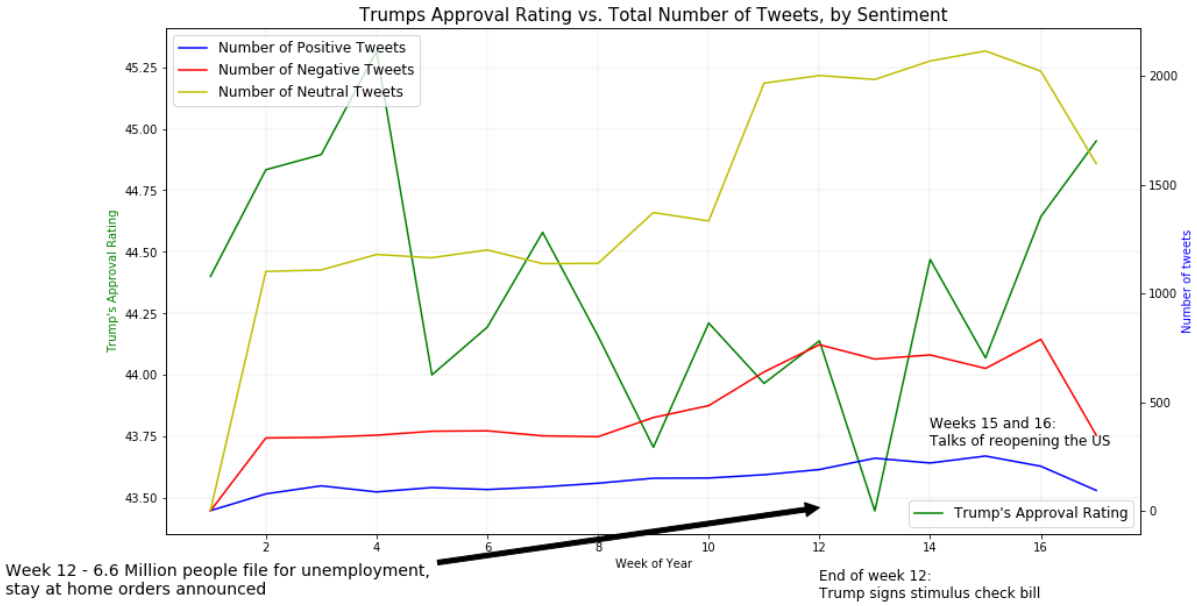



## Number of Total Tweets by Sentiment

These give some interesting insights into how these figures may be moving inversely (more on this in the discussion later), but let's see how this plot changes when we break it down by sentiment.

In [178]:

```
import numpy as np
import matplotlib.pyplot as plt
x = approval_df.groupby('week_bins')['approve'].mean().index
y1 = approval_df.groupby('week_bins')['approve'].mean()
y2 = arrpos
y3 = arrneg
y4 = arrneut

fig, ax1 = plt.subplots(figsize=(15,8))

ax1.set_title('Trumps Approval Rating vs. Total Number of Tweets, by Sentimen
t', fontsize=15)
ax1.grid(color='grey', linestyle='-', linewidth=0.25, alpha=0.5)

#duplicate ax1
ax2 = ax1.twinx()

#ax1 will plot Trump's approval rating, ax2 will plot number of tweets
ax1.plot(x, y1, 'g-')
ax2.plot(x, y2, 'b-')
ax2.plot(x, y3, 'r-')
ax2.plot(x, y4, 'y-')


# set legends
ax1.legend(["Trump's Approval Rating"], fontsize=12, loc=4)
ax2.legend(["Number of Positive Tweets", "Number of Negative Tweets", "Number
 of Neutral Tweets"], fontsize=12, loc=2)

plt.text(14,300, """Weeks 15 and 16:
Talks of reopening the US""", fontsize = 12)

plt.annotate("""Week 12 - 6.6 Million people file for unemployment,
stay at home orders announced""",
             xy = (12,15), xytext = (2,15), textcoords = ('figure points'),
             arrowprops=dict(facecolor='black'), fontsize = 14)

plt.text(12,-400, """End of week 12:
Trump signs stimulus check bill""", fontsize = 12)




ax1.set_xlabel('Week of Year')
ax1.set_ylabel("Trump's Approval Rating", color='g')
ax2.set_ylabel("Number of tweets", color='b')
#ax.legend((y1, y2, y3), ('label1', 'label2', 'label3'))


plt.show()
```

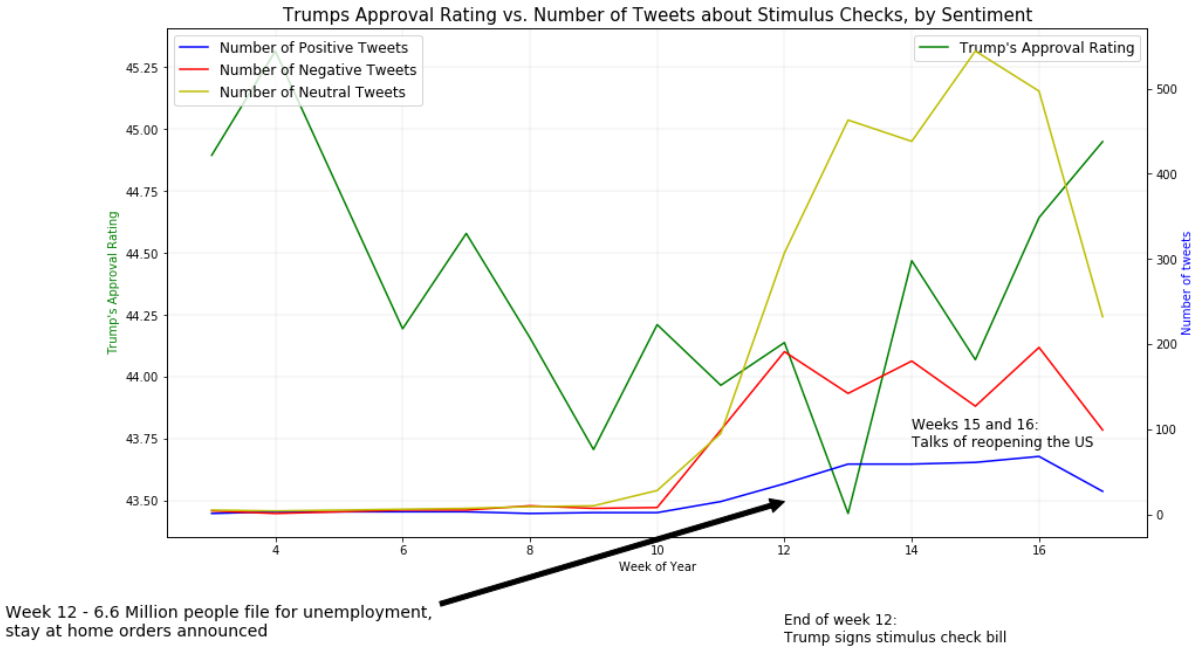



## Number of Stimulus Check Tweets by Sentiment

Let's take our positive, negative, and neutral tweets and search for the ones where stimulus check is mentioned

In [179]:

```
# import numpy as np
import matplotlib.pyplot as plt
x = approval_temp_x
y1 = approval_temp_y
y2 = stimuluscheck_pos_df.groupby('week_bins')['sentimentLabel'].count()
y3 = stimcheckneg
y4 = stimcheckneut

fig, ax1 = plt.subplots(figsize=(15,8))

ax1.set_title('Trumps Approval Rating vs. Number of Tweets about Stimulus Chec
ks, by Sentiment', fontsize=15)
ax1.grid(color='grey', linestyle='-', linewidth=0.25, alpha=0.5)

#duplicate ax1
ax2 = ax1.twinx()

#ax1 will plot Trump's approval rating, ax2 will plot number of tweets
ax1.plot(x, y1, 'g-')
ax2.plot(x, y2, 'b-') # POSITIVE TWEETS
ax2.plot(x, y3, 'r-') # NEGATIVE TWEETS
ax2.plot(x, y4, 'y-') # NEUTRAL TWEETS

# set legends
ax1.legend(["Trump's Approval Rating"], fontsize=12, loc=1)
ax2.legend(["Number of Positive Tweets", "Number of Negative Tweets", "Number
 of Neutral Tweets"], fontsize=12, loc=2)

plt.text(14,80, """Weeks 15 and 16:
Talks of reopening the US""", fontsize = 12)

plt.annotate("""Week 12 - 6.6 Million people file for unemployment,
stay at home orders announced""",
             xy = (12,15), xytext = (2,15), textcoords = ('figure points'),
             arrowprops=dict(facecolor='black'), fontsize = 14)

plt.text(12,-150, """End of week 12:
Trump signs stimulus check bill""", fontsize = 12)


ax1.set_xlabel('Week of Year')
ax1.set_ylabel("Trump's Approval Rating", color='g')
ax2.set_ylabel("Number of tweets", color='b')

plt.show()
```

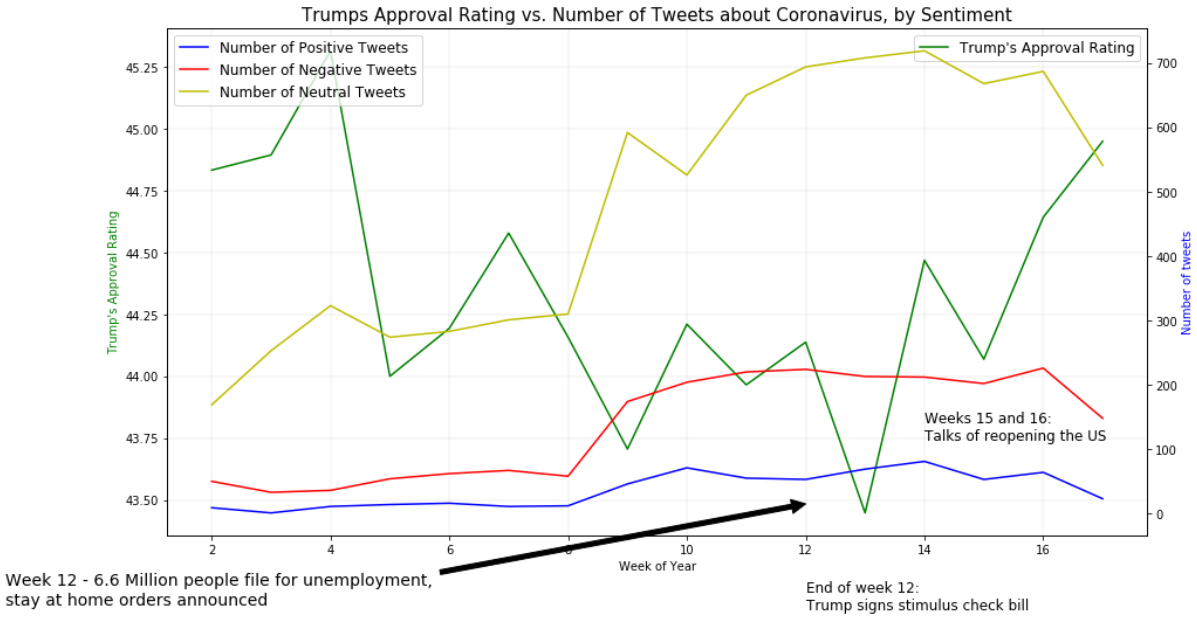



## Number of Coronavirus Tweets by Sentiment

Let's take our positive, negative, and neutral tweets and search for the ones where stimulus check is mentioned

In [180]:

```
# import numpy as np
import matplotlib.pyplot as plt
x = corona_temp_x
y1 = corona_temp_y
y2 = corona_pos_df.groupby('week_bins')['sentimentLabel'].count()
y3 = corona_neg_df.groupby('week_bins')['sentimentLabel'].count()
y4 = corona_neut_df.groupby('week_bins')['sentimentLabel'].count()

fig, ax1 = plt.subplots(figsize=(15,8))

ax1.set_title('Trumps Approval Rating vs. Number of Tweets about Coronavirus,
 by Sentiment', fontsize=15)
ax1.grid(color='grey', linestyle='-', linewidth=0.25, alpha=0.5)

#duplicate ax1
ax2 = ax1.twinx()

#ax1 will plot Trump's approval rating, ax2 will plot number of tweets
ax1.plot(x, y1, 'g-')
ax2.plot(x, y2, 'b-') # POSITIVE TWEETS
ax2.plot(x, y3, 'r-') # NEGATIVE TWEETS
ax2.plot(x, y4, 'y-') # NEUTRAL TWEETS

# set legends
ax1.legend(["Trump's Approval Rating"], fontsize=12, loc=1)
ax2.legend(["Number of Positive Tweets", "Number of Negative Tweets", "Number
 of Neutral Tweets"], fontsize=12, loc=2)

ax1.set_xlabel('Week of Year')
ax1.set_ylabel("Trump's Approval Rating", color='g')
ax2.set_ylabel("Number of tweets", color='b')



plt.text(14,115, """Weeks 15 and 16:
Talks of reopening the US""", fontsize = 12)

plt.annotate("""Week 12 - 6.6 Million people file for unemployment,
stay at home orders announced""",
             xy = (12,15), xytext = (2,15), textcoords = ('figure points'),
             arrowprops=dict(facecolor='black'), fontsize = 14)

plt.text(12,-150, """End of week 12:
Trump signs stimulus check bill""", fontsize = 12)



plt.show()
```




Further analysis of graph will be discussed in the "discussion" and "conclusion" sections

# Part 2: Predicting the 2020 Election

## Description Of The Data - Part 2

## Scraper(s) Used

For this part of the project, we scraped for the same keywords as before, but this time we used the API scraper to get more info for users on current tweets.

In [181]:

```
from twython import TwythonStreamer
import csv
import json
import codecs
import time
from random import seed
from random import randint
# seed random number generator
seed(1)

tweets_filename = "twitter_output_Trump"

consumer_key = "ND4wRUZRSgRS1NBtb9vRbm96v"
consumer_secret = "z0zxb6QObq4AUU2bA8BBclnIZbDr89vShniSZ9NjMWHUaIJ2AK"
access_token = "1250793114274598913-1mA5LzmdeGsu7PvTD1QpoZymfUPrPl"
access_token_secret = "cqNiCEcr5OsbS5G1t3uAGhZhqabgeu7XVHZluQiJIBVnf"

tweetabbr = []

# In this session we are using the Twitter IDs to gather tweets from those acc
ounts.
AllPDs = ['561106229', '34296669', '974277346252423169', '121222566', '9915253
50', '378424739',
          '35871927', '304847225']
# Filter out unwanted data for the CSV file. We are saving the entire JSON to
 a seperate file should we need more data.
def process_tweet(tweet):
    d = {}
    d['hashtags'] = [hashtag['text'] for hashtag in tweet['entities']['hashtag
s']]
    d['id'] = tweet['id']
    d['text'] = tweet['text']
    d['name'] = tweet['user']['name']
    d['user'] = tweet['user']['screen_name']
    d['user_loc'] = tweet['user']['location']
    d['user_desc'] = tweet['user']['description']
    d['user_followers'] = tweet['user']['followers_count']
    d['user_friends'] = tweet['user']['friends_count']
    d['user_listed'] = tweet['user']['listed_count']
    d['user_created'] = tweet['user']['created_at']
    d['user_favs'] = tweet['user']['favourites_count']
    d['user_statuses'] = tweet['user']['statuses_count']

    return d


# Create a class that inherits TwythonStreamer
class MyStreamer(TwythonStreamer):

    # Received data
    def on_success(self, data):

        # Save full JSON to file
        # TODO : save properly so we can load later directly
        # A tweet JSON record per line
        with open(f'{tweets_filename}.json', 'a') as jsonfile:
```

```
            json.dump(data, jsonfile)
            jsonfile.write("\n")

        # Only save tweets in English
        if data['lang'] == 'en':
            tweet_data = process_tweet(data)
            self.save_to_csv(tweet_data)

    # Problem with the API
    def on_error(self, status_code, data):
        print(status_code, data)
        self.disconnect()

    # Save each tweet to csv file
    def save_to_csv(self, tweet):
        with open(f'{tweets_filename}.csv', 'a', encoding="utf8") as file:
            writer = csv.writer(file)
            writer.writerow(list(tweet.values()))

while True:
    try:
        # Instantiate from our streaming class
        stream = MyStreamer(consumer_key, consumer_secret,
                    access_token, access_token_secret)

        # Start the stream - this would capture tweets generated by specific a
ccounts - ex. Harley Davidson's account
        # There are online tools to get the account number using the @ account
name.
        # stream.statuses.filter(follow=17169239) #Track uses comma separated
 list

        # Start the stream - this would capture tweets generated by these acco
unts
        #stream.statuses.filter(follow=AllPDs) #Track uses comma separated lis
t

        # Start the stream - This stream looks for specific terms (mentions)
        #stream.statuses.filter(track='@VASenate2018,@MariaCantwell,@Susan_Hut
ch,') #Track uses comma separated list
        stream.statuses.filter(track='Trump', 'Stimulus Check','$1200', 'covid
19', 'coronavirus','#trump','#stimuluscheck', '#money' )
#track="Trump, Stimulus Check,stimulus check,trump,government,Government,Repub
lican,owner,owner,Mayor,State Rep,democrat,#trump,#stimuluscheck,#money,#repub
lican,#COVID19,financial crisis"#track="Trump, Stimulus Check,stimulus check,t
rump,government,Government,Republican,owner,owner,Mayor,State Rep,democrat,#tr
ump,#stimuluscheck,#money,#republican,#COVID19,financial crisis"
    except (KeyboardInterrupt):
        print("Exiting")
        break
    except Exception as e:
        print("error - sleeping " + str(e))
        time.sleep(randint(30, 90)) #suspends (waits) execution of the current
thread for a given number of seconds
        continue
```

```
  File "<ipython-input-181-c91607ce482b>", line 87
    stream.statuses.filter(track='Trump', 'Stimulus Check','$1200', 'covid1
9', 'coronavirus','#trump','#stimuluscheck', '#money' )
                                                      ^
SyntaxError: positional argument follows keyword argument
```

# Data Description

Our second analysis involves data scraped from twitter API. This allows us to work more with user data to produce our simulation.

- **We scraped the following tweet characteristics via the API (second set of data)**:
  - Hashtags
  - Tweet ID
  - Tweet text
  - User's display name
  - Username
  - User's location
  - User's description
  - Followers
  - Friends
  - Lists
  - Account creation date
  - Favorites
  - User total tweets
- **We then added the following**:
  - filtered_text
  - retweet_flags
  - Polarity_Score
  - SubjectivityScore
  - sentimentLabel

## Data Processing Tasks

For our second set of data, we will first have to perform sentiment analysis. The first step in this was to get the sentiment bins. We then saw how many neutral values there were, so we did a further analysis on those neutral values to then filter then again by their user_description.

## When And How Long You Scraped Twitter

For this part of the project, we scraped Twitter for about 8 hours continuously on Thursday April 30 2020.

# Load in Data

In [182]:
```
import pandas as pd
import numpy as np

df= pd.read_csv("Group_2_Phase_4_twitter_output_Trump.csv")
df.columns= ['Hashtags','ID','Tweet_Text','Name','Username','User_Location','User_Description','Followers','Friends','Lists','Account_Creation_Date','Favorites','User_Total_Tweets']
```

### Drop Duplicates

In [183]:
```
length  = len(df)
new_length = len(df.drop_duplicates())
print(f" The current length of our Dataframe is {length} records")
print(f" The length of our Dataframe after dropping duplicates is {new_length} records" )
```

```
 The current length of our Dataframe is 28861 records
 The length of our Dataframe after dropping duplicates is 28857 records
```

# Simulation Data Cleaning/ Sorting

Importing libraries and functions given to us from Professor Shanahan

In [184]:
```
# Remove URLs and Stop Words
df['filtered_text'] = df.Tweet_Text.apply(preprocess_tweet_text)

#retweet
df['retweet_flags'] = df.Tweet_Text.str.startswith('RT')

# TODO Add code analyse URLs, their domains, categories etc.
```

In [185]:
```
# deal with nots
!pip install TextBlob
```

```
Requirement already satisfied: TextBlob in /usr/local/lib/python3.6/site-packages (0.15.3)
Requirement already satisfied: nltk>=3.1 in /usr/local/lib/python3.6/site-packages (from TextBlob) (3.4.5)
Requirement already satisfied: six in /usr/local/lib/python3.6/site-packages (from nltk>=3.1->TextBlob) (1.14.0)
```

In [186]:
```
from textblob import TextBlob
df['PolarityScore'] = df.filtered_text.apply(lambda txt:  TextBlob(txt).polarity)
df['SubjectivityScore'] = df.filtered_text.apply(lambda txt:  TextBlob(txt).subjectivity)
```

In [187]:
```
def sentiment_bins(data):
    if data <= -0.5:
        grouping = 'Strong-Negative'
    elif data > -0.5 and data < 0.0:
        grouping = 'Mild-Negative'
    elif data > 0.0 and data < 0.5:
        grouping = 'Mild Positive'
    elif data >=0.5:
        grouping = 'Strong-Positive'
    else:
        grouping = 'Neutral'
    return grouping

df['sentimentLabel'] = df['PolarityScore'].apply(sentiment_bins)
```

In [188]:
```
# user locations
df.User_Location.value_counts(dropna = True)
```

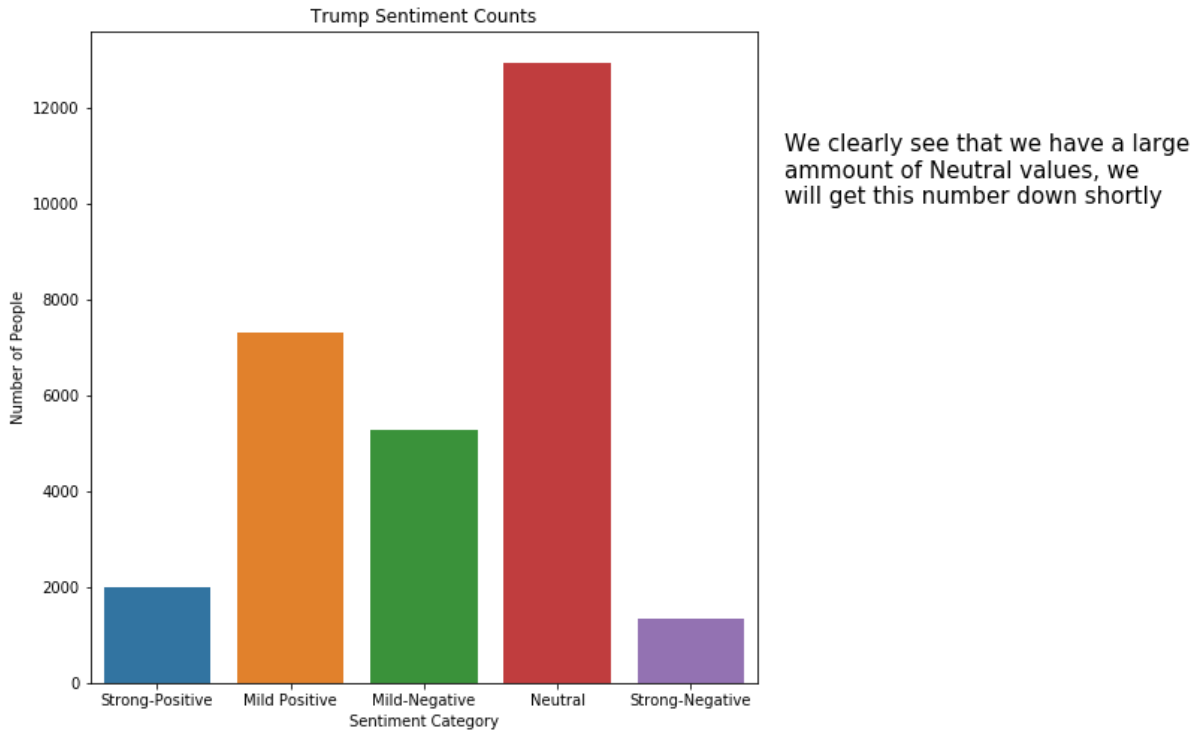
Out[188]:
```
United States                       716
USA                                 253
California, USA                     253
Texas, USA                          172
India                               148
                                   ...
Time is a man made concept!!!         1
From NYC. Live in La Jolla, CA        1
Arizona USA                           1
Paddington, London                    1
society                               1
Name: User_Location, Length: 8696, dtype: int64
```

# Visual EDA- Part 2

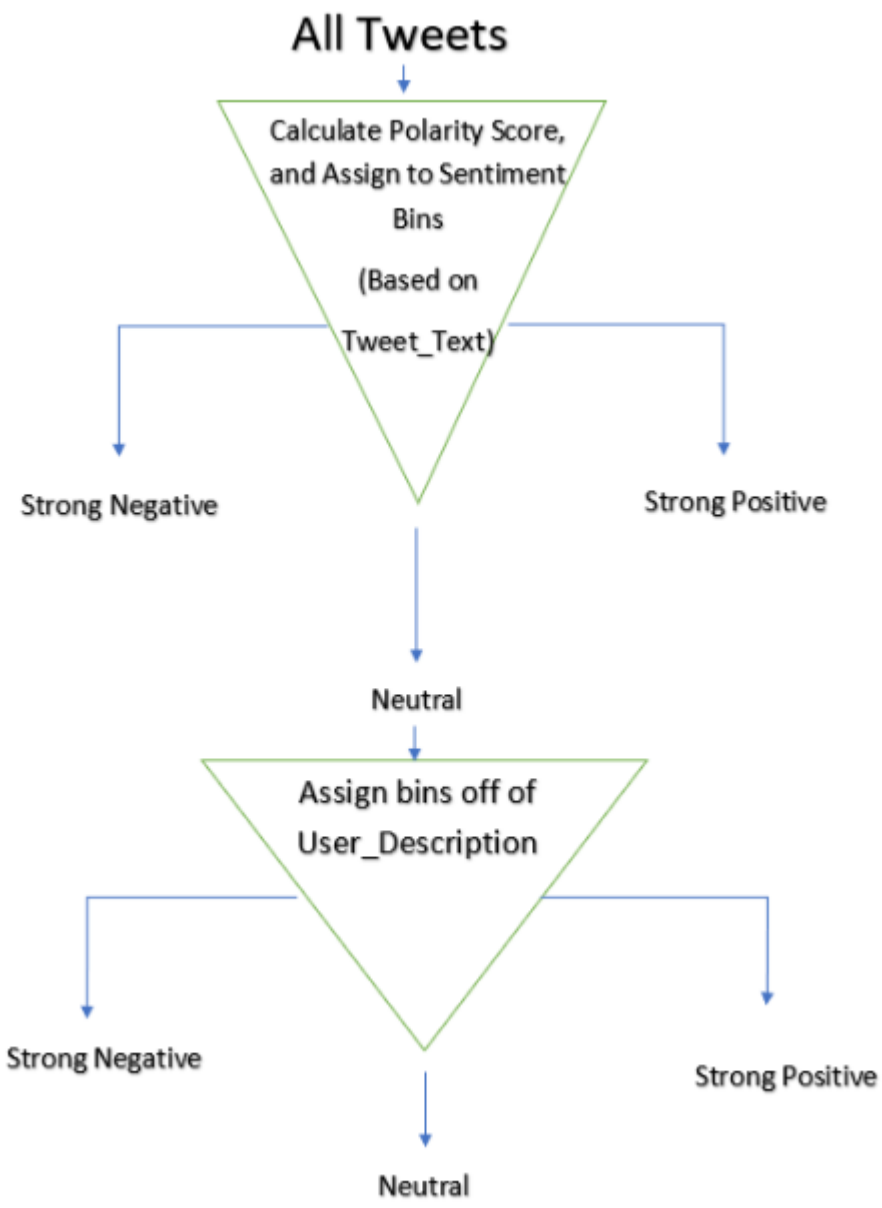
## Further Filtering Neutral Values

In [189]:
```
plt.figure(figsize = (8,8))
ax = sns.countplot(x = "sentimentLabel", data = df)
ax.set(title = "Trump Sentiment Counts", xlabel = "Sentiment Category", ylabel
= "Number of People")
plt.text(4.7, 10000,"We clearly see that we have a large\nammount of Neutral v
alues, we\nwill get this number down shortly", fontsize = 15)
```

Out[189]:
```
Text(4.7, 10000, 'We clearly see that we have a large\nammount of Neutral val
ues, we\nwill get this number down shortly')
```




From a basic level, this is how we are filtering neutral values.


All Tweets
Calculate Polarity Score, and Assign to Sentiment Bins
(Based on Tweet_Text)
Strong Negative
Strong Positive
Neutral
Assign bins off of User_Description
Strong Negative
Strong Positive
Neutral

## Creating the Neutral Dataframe

In [190]:
```
Neutral= df[df['sentimentLabel'].str.contains("Neutral", flags=re.IGNORECASE)]
#Creating a datafrmae of all the Neutral tweets
Neutral['User_Description'] = Neutral.User_Description.fillna('.') # More data
cleaning
```

```
/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCo
pyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports u
ntil
```

So here we are taking our dataframe of Neutral tweets and then calculating a polarity score for them based off of User_Description to further filter them to be able to get rid of the number of Neutral values.

After looking at some of the User_Descriptions we found that they actually had value as some of them would explicitly state their political affiliation (ex. Patriot Pro-Trump Christian English). The index for this record was 3.

In [191]:
```
Neutral['PolarityScore'] = Neutral.User_Description.apply(lambda txt:  TextBlo
b(txt).polarity)
Neutral['SubjectivityScore'] = Neutral.User_Description.apply(lambda txt:  Tex
tBlob(txt).subjectivity)
#Actually creating the Polarity and Subjectivity scores
Neutral['sentimentLabel'] = Neutral['PolarityScore'].apply(sentiment_bins)
#Now putting them into bins
```

```
/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:1: SettingWithCo
pyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:2: SettingWithCo
pyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:4: SettingWithCo
pyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

We now have the neutral values with an assigned sentiment label, which is based off of the user_description!

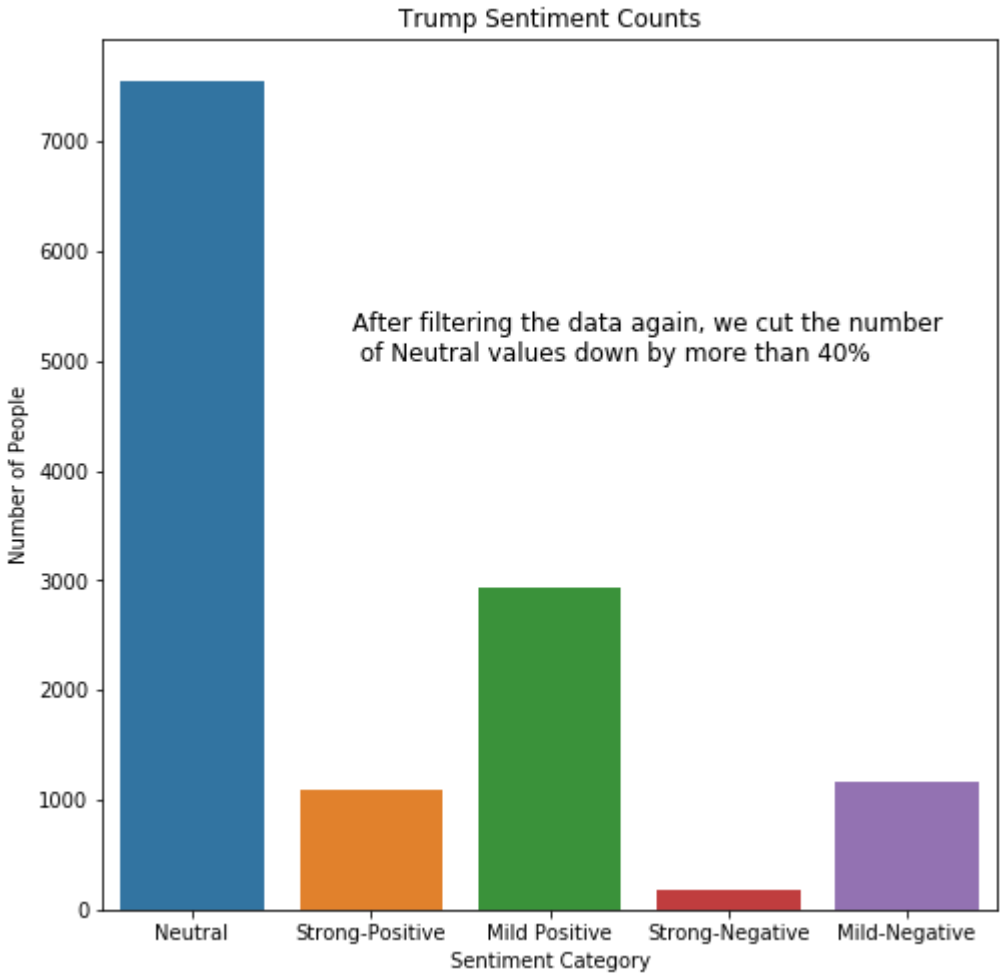
In [192]:
```
Neutral.head()
```

Out[192]:

| | Hashtags | ID | Tweet_Text | Name | Username | User_Loca |
|---|---|---|---|---|---|---|
| **3** | [] | 1254972505367994368 | RT @RedWingGrips: BREAKING: President Trump ha... | Proud American | ProudQchick | United S |
| **4** | [] | 1254972505497808897 | RT @mog7546: Health experts from the Trump, Ob... | Donald Putin is a TRAITOR | GalindoSherry | Saskatche Ca |
| **5** | [] | 1254972505544105984 | RT @costareports: “U.S. intelligence agencies ... | Joni A. Caufield | TL267sister | |
| **9** | [] | 1254972505661616129 | RT @soledadobrien: That ‘somebody’ is you, babe. | Honey | TheHoneyPirate | |
| **10** | [] | 1254972505837764610 | Trump Blames Plummeting Poll Numbers on People... | MADgenius_123 | MADgenius_123 | |

In [193]:
```
plt.figure(figsize = (8,8))
ax = sns.countplot(x = "sentimentLabel", data = Neutral)
ax.set(title = "Trump Sentiment Counts", xlabel = "Sentiment Category", ylabel
= "Number of People")
plt.text(0.9,5000,"After filtering the data again, we cut the number\n of Neut
ral values down by more than 40%", fontsize = 12)
```

Out[193]:
```
Text(0.9, 5000, 'After filtering the data again, we cut the number\n of Neutr
al values down by more than 40%')
```




# EDA Part 2

## 2020 Election Simulation

This section includes an elaborate simulation of the 2020 Election based off of the Top 10 states (ranked by number of electoral votes). We then looked at the outcomes of the previous election to determine if a state would be a swing state or not. If the margin of victory was lower than 5% then we determined that the state would be defined as a "Swing State" (Meaning that this state could "swing" sides and belong to the other party in the upcoming election. This also led to theory that on election day, a swing state could be such a close race, and could ultimately be a 50/50 chance for each party (This theory is also further explained in the metric). In addition this is only a small simulation of the actual election, and is determinig the "winner" based off of only the top 10 states for simplicity.

In [194]:
```
topStates = pd.read_csv("Group_2_Phase_4_Top10States.csv") # The states with the most Electoral votes
topStates #A BLUE STATE= WON BY THE DEMOCRATIC PARTY AND RED MEANING IT WAS WON BY THE REUPUBLICAN PARTY
```

Out[194]:

| | State | Electoral Votes | Difference in % of Votes | Swing State | 2016 Status |
|---|---|---|---|---|---|
| **0** | California | 53 | 28.80% | No | Blue |
| **1** | Texas | 36 | 9.20% | No | Red |
| **2** | New York | 27 | 21.30% | No | Blue |
| **3** | Florida | 27 | 1.30% | Yes | Red |
| **4** | Pennsylvania | 18 | 1.20% | Yes | Red |
| **5** | Illinois | 18 | 16% | No | Blue |
| **6** | Ohio | 16 | 8.60% | No | Red |
| **7** | Michigan | 14 | 0.30% | Yes | Red |
| **8** | Georgia | 14 | 5.70% | No | Red |
| **9** | North Carolina | 13 | 3.80% | Yes | Red |

## How the simulation actually works

In summary, the simulation will follow these steps to determine the winner of the 2020 Election, based off of sentiment and location.

- PLEASE KEEP IN MIND THAT THIS IS FROM TRUMP'S PERSPECTIVE (Negative means Not for Trump, Positive means for Trump)

1. Add up all of the positive and negative tweets for the specified state from the filtered Neutral DataFrame based off of their new label.
2. Do the exact same for the original DataFrame to get a Positive and Negative score
3. Take the number of remaining Neutral values for each state
4. It is now determined if a state is a swing state or not
5. If the state is not a swing state, then the remaining neutral values are given the party that had won the state in the previous election. But if the state is indeed a swing state, it enters into a for loop, where based off of our theory that it's almost a 50/50 chance of winning one of those states, a random number is given. If the random number is even, then those states neutral votes are given to Joe Biden, but if that number comes out to odd, then those votes are given to Donald Trump.
6. Then based off of that determination, we calculate who wins the state based off of who got the most votes.

Below is a screenshot for just one state, this then replicated for every state in the Top 10

```
Neutral['User_Location'] = df.User_Location.fillna("")
#Creating positive and negative counts for the filtered Neutral dataframe by state
FLN = Neutral[Neutral['User_Location'].str.contains("Florida|FL", flags=re.IGNORECASE)]
FLN1= FLN[FLN['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNORECASE)]
FLN2= FLN[FLN['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNORECASE)]
FLN3 = FLN[FLN['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORECASE)]
FLN4 = FLN[FLN['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNORECASE)]
NegativeN = len(FLN1)+len(FLN2)
PositiveN = len(FLN3)+len(FLN4)
#Now creating positive and negative counts for the original dataframe by state
df['User_Location'] = df.User_Location.fillna('')
FL = df[df['User_Location'].str.contains("FLORIDA|FL", flags=re.IGNORECASE)]
FL1= FL[FL['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNORECASE)]
FL2= FL[FL['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNORECASE)]
FL3 = FL[FL['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORECASE)]
FL4 = FL[FL['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNORECASE)]
Negative = len(FL1)+len(FL2)
Positive = len(FL3)+len(FL4)
Only_Neutral = FLN[FLN['sentimentLabel'].str.contains("Neutral", flags=re.IGNORECASE)]
Neutral_Final = len(Only_Neutral)
#This for loop is entered becuase the state is a swing state, would just assign neutral values based off of the results
#from the election before
import numpy as np
random = np.random.randint(low = 1, high = 10)
if (random%2) == 0:
        Negative_Final = Negative + NegativeN + Neutral_Final
        Positive_Final = Positive + PositiveN
else:
    Positive_Final = Positive + PositiveN + Neutral_Final
    Negative_Final = Negative + NegativeN
if Negative_Final > Positive_Final:
    Biden_vote_count = Biden_vote_count+27
    print("Biden Wins Florida")
    print(f"Biden Vote Count: {Biden_vote_count}")
    print('-'*50)
#Updating the list to later analyze in our final DataFrame
    trump_win.append(0)
    biden_win.append(1)
else:
    Trump_vote_count = Trump_vote_count+27
    print("Trump Wins Florida")
    print(f"Trump Vote Count: {Trump_vote_count}")
    print('-'*50)
    trump_win.append(1)
    biden_win.append(0)
```

## The Actual Simulation

Please note this this is a live simulation, so each time this cell is ran, it will produce different outputs. For our analysis we used the first output we got from this simulation.

In [195]:

```
Biden_vote_count = 0
Trump_vote_count = 0
trump_win = []
biden_win = []
# CALIFORNIA-------------------------------------------------------------------
------------------------------
Neutral['User_Location'] = df.User_Location.fillna("")
CALN = Neutral[Neutral['User_Location'].str.contains("California|CA", flags=re
.IGNORECASE)]
CALN1= CALN[CALN['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNO
RECASE)]
CALN2= CALN[CALN['sentimentLabel'].str.contains("Strong-Negative", flags=re.IG
NORECASE)]
CALN3 = CALN[CALN['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGN
ORECASE)]
CALN4 = CALN[CALN['sentimentLabel'].str.contains("Strong-Positive", flags=re.I
GNORECASE)]
NegativeN = len(CALN1)+len(CALN2)
PositiveN = len(CALN3)+len(CALN4)
df['User_Location'] = df.User_Location.fillna('')
CA = df[df['User_Location'].str.contains("California|CA", flags=re.IGNORECASE
)]
CAL1= CA[CA['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNORECAS
E)]
CAL2= CA[CA['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNOREC
ASE)]
CAL3 = CA[CA['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORECA
SE)]
CAL4 = CA[CA['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNORE
CASE)]
Negative = len(CAL1)+len(CAL2)
Positive = len(CAL3)+len(CAL4)
Only_Neutral = CALN[CALN['sentimentLabel'].str.contains("Neutral", flags=re.IG
NORECASE)]
Neutral_Final = len(Only_Neutral)
Positive_Final = Positive + PositiveN
Negative_Final = Negative + NegativeN+Neutral_Final
if Negative_Final > Positive_Final:
    Biden_vote_count = Biden_vote_count+53
    print("Biden Wins California")
    print(f"Biden Vote Count: {Biden_vote_count}")
    print('-'*50)
    trump_win.append(0)
    biden_win.append(1)
else:
    Trump_vote_count = Trump_vote_count+53
    print("Trump Wins California")
    print(f"Trump Vote Count: {Trump_vote_count}")
    trump_win.append(1)
    biden_win.append(0)
#TEXAS-------------------------------------------------------------------------
---------------------
Neutral['User_Location'] = df.User_Location.fillna("")
TXN = Neutral[Neutral['User_Location'].str.contains("Texas|TX", flags=re.IGNOR
ECASE)]
TXN1= TXN[TXN['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNOREC
```

```
ASE)]
TXN2= TXN[TXN['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNOR
ECASE)]
TXN3 = TXN[TXN['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORE
CASE)]
TXN4 = TXN[TXN['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNO
RECASE)]
NegativeN = len(TXN1)+len(TXN2)
PositiveN = len(TXN3)+len(TXN4)
df['User_Location'] = df.User_Location.fillna('')
Neutral['User_Location'] = df.User_Location.fillna("")
TX = df[df['User_Location'].str.contains("Texas|TX", flags=re.IGNORECASE)]
TX1= TX[TX['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNORECASE
)]
TX2= TX[TX['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNORECA
SE)]
TX3 = TX[TX['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORECAS
E)]
TX4 = TX[TX['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNOREC
ASE)]
Negative = len(TX1)+len(TX2)
Positive = len(TX3)+len(TX4)
Only_Neutral = TXN[TXN['sentimentLabel'].str.contains("Neutral", flags=re.IGNO
RECASE)]
Neutral_Final = len(Only_Neutral)
Positive_Final = Positive + PositiveN+Neutral_Final
Negative_Final = Negative + NegativeN
Neutral_Final = len(Only_Neutral)
if Negative_Final > Positive_Final:
    Biden_vote_count = Biden_vote_count+36
    print("Biden Wins Texas")
    print(f"Biden Vote Count: {Biden_vote_count}")
    print('-'*50)
    trump_win.append(0)
    biden_win.append(1)
else:
    Trump_vote_count = Trump_vote_count+36
    print("Trump Wins Texas")
    print(f"Trump Vote Count: {Trump_vote_count}")
    print('-'*50)
    trump_win.append(1)
    biden_win.append(0)
# NEW YORK-----------------------------------------------------------------
---------------------------------
Neutral['User_Location'] = df.User_Location.fillna("")
NYN = Neutral[Neutral['User_Location'].str.contains("New York|NY", flags=re.IG
NORECASE)]
NYN1= NYN[NYN['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNOREC
ASE)]
NYN2= NYN[NYN['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNOR
ECASE)]
NYN3 = NYN[NYN['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORE
CASE)]
NYN4 = NYN[NYN['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNO
RECASE)]
NegativeN = len(NYN1)+len(NYN2)
PositiveN = len(NYN3)+len(NYN4)
```

```
df['User_Location'] = df.User_Location.fillna('')
NY = df[df['User_Location'].str.contains("New York|NY", flags=re.IGNORECASE)]
NY1= NY[NY['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNORECASE
)]
NY2= NY[NY['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNORECA
SE)]
NY3 = NY[NY['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORECAS
E)]
NY4 = NY[NY['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNOREC
ASE)]
Negative = len(NY1)+len(NY2)
Positive = len(NY3)+len(NY4)
Only_Neutral = NYN[NYN['sentimentLabel'].str.contains("Neutral", flags=re.IGNO
RECASE)]
Neutral_Final = len(Only_Neutral)
Positive_Final = Positive + PositiveN
Negative_Final = Negative + NegativeN+Neutral_Final
if Negative_Final > Positive_Final:
    Biden_vote_count = Biden_vote_count+27
    print("Biden Wins New York")
    print(f"Biden Vote Count: {Biden_vote_count}")
    print('-'*50)
    trump_win.append(0)
    biden_win.append(1)
else:
    Trump_vote_count = Trump_vote_count+27
    print("Trump Wins New York")
    print(f"Trump Vote Count: {Trump_vote_count}")
    print('-'*50)
    trump_win.append(1)
    biden_win.append(0)
# FLORIDA-----------------------------------------------------------------------
----------------------------
Neutral['User_Location'] = df.User_Location.fillna("")
FLN = Neutral[Neutral['User_Location'].str.contains("Florida|FL", flags=re.IGN
ORECASE)]
FLN1= FLN[FLN['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNOREC
ASE)]
FLN2= FLN[FLN['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNOR
ECASE)]
FLN3 = FLN[FLN['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORE
CASE)]
FLN4 = FLN[FLN['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNO
RECASE)]
NegativeN = len(FLN1)+len(FLN2)
PositiveN = len(FLN3)+len(FLN4)
df['User_Location'] = df.User_Location.fillna('')
FL = df[df['User_Location'].str.contains("FLORIDA|FL", flags=re.IGNORECASE)]
FL1= FL[FL['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNORECASE
)]
FL2= FL[FL['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNORECA
SE)]
FL3 = FL[FL['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORECAS
E)]
FL4 = FL[FL['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNOREC
ASE)]
Negative = len(FL1)+len(FL2)
```

```
Positive = len(FL3)+len(FL4)
Only_Neutral = FLN[FLN['sentimentLabel'].str.contains("Neutral", flags=re.IGNO
RECASE)]
Neutral_Final = len(Only_Neutral)
import numpy as np
random = np.random.randint(low = 1, high = 10)
if (random%2) == 0:
        Negative_Final = Negative + NegativeN + Neutral_Final
        Positive_Final = Positive + PositiveN
else:
    Positive_Final = Positive + PositiveN + Neutral_Final
    Negative_Final = Negative + NegativeN
if Negative_Final > Positive_Final:
    Biden_vote_count = Biden_vote_count+27
    print("Biden Wins Florida")
    print(f"Biden Vote Count: {Biden_vote_count}")
    print('-'*50)
    trump_win.append(0)
    biden_win.append(1)
else:
    Trump_vote_count = Trump_vote_count+27
    print("Trump Wins Florida")
    print(f"Trump Vote Count: {Trump_vote_count}")
    print('-'*50)
    trump_win.append(1)
    biden_win.append(0)
#Pennsylvania-----------------------------------------------------------------
----------------------------
Neutral['User_Location'] = df.User_Location.fillna("")
PNN = Neutral[Neutral['User_Location'].str.contains("Pennsylvania|PN", flags=r
e.IGNORECASE)]
PNN1= PNN[PNN['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNOREC
ASE)]
PNN2= PNN[PNN['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNOR
ECASE)]
PNN3 = PNN[PNN['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORE
CASE)]
PNN4 = PNN[PNN['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNO
RECASE)]
NegativeN = len(PNN1)+len(PNN2)
PositiveN = len(PNN3)+len(PNN4)
df['User_Location'] = df.User_Location.fillna('')
PN = df[df['User_Location'].str.contains("Pennsylvania|PA", flags=re.IGNORECAS
E)]
PN1= PN[PN['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNORECASE
)]
PN2= PN[PN['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNORECA
SE)]
PN3 = PN[PN['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORECAS
E)]
PN4 = PN[PN['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNOREC
ASE)]
Negative = len(PN1)+len(PN2)
Positive = len(PN3)+len(PN4)
Only_Neutral = PN[PN['sentimentLabel'].str.contains("Neutral", flags=re.IGNORE
CASE)]
Neutral_Final = len(Only_Neutral)
```

```
import numpy as np
random = np.random.randint(low = 1, high = 10)
if (random%2) == 0:
        Negative_Final = Negative + NegativeN + Neutral_Final
        Positive_Final = Positive + PositiveN
else:
    Positive_Final = Positive + PositiveN + Neutral_Final
    Negative_Final = Negative + NegativeN
if Negative_Final > Positive_Final:
    Biden_vote_count = Biden_vote_count+18
    print("Biden Wins Pennsylvania")
    print(f"Biden Vote Count: {Biden_vote_count}")
    print('-'*50)
    trump_win.append(0)
    biden_win.append(1)
else:
    Trump_vote_count = Trump_vote_count+18
    print("Trump Wins Pennsylvania")
    print(f"Trump Vote Count: {Trump_vote_count}")
    print('-'*50)
    trump_win.append(1)
    biden_win.append(0)
#ILLINOIS-----------------------------------------------------------------------
-------------------------------------------
Neutral['User_Location'] = df.User_Location.fillna("")
ILN = Neutral[Neutral['User_Location'].str.contains("Illinois|IL", flags=re.IG
NORECASE)]
ILN1= ILN[ILN['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNOREC
ASE)]
ILN2= ILN[ILN['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNOR
ECASE)]
ILN3 = ILN[ILN['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORE
CASE)]
ILN4 = ILN[ILN['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNO
RECASE)]
NegativeN = len(ILN1)+len(ILN2)
PositiveN = len(ILN3)+len(ILN4)
df['User_Location'] = df.User_Location.fillna('')
IL = df[df['User_Location'].str.contains("Illinois|IL", flags=re.IGNORECASE)]
IL1= IL[IL['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNORECASE
)]
IL2= IL[IL['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNORECA
SE)]
IL3 = IL[IL['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORECAS
E)]
IL4 = IL[IL['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNOREC
ASE)]
Negative = len(IL1)+len(IL2)
Positive = len(IL3)+len(IL4)
Only_Neutral = ILN[ILN['sentimentLabel'].str.contains("Neutral", flags=re.IGNO
RECASE)]
Neutral_Final = len(Only_Neutral)
Positive_Final = Positive + PositiveN
Negative_Final = Negative + NegativeN+Neutral_Final
if Negative_Final > Positive_Final:
    Biden_vote_count = Biden_vote_count+18
    print("Biden Wins Illinois")
```

```
    print(f"Biden Vote Count: {Biden_vote_count}")
    print('-'*50)
    trump_win.append(0)
    biden_win.append(1)
else:
    Trump_vote_count = Trump_vote_count+18
    print("Trump Wins Illinois")
    print(f"Trump Vote Count: {Trump_vote_count}")
    trump_win.append(1)
    biden_win.append(0)
#OHIO-------------------------------------------------------------------------
---------------------------------------
Neutral['User_Location'] = df.User_Location.fillna("")
OHN = Neutral[Neutral['User_Location'].str.contains("Ohio|OH", flags=re.IGNORE
CASE)]
OHN1= OHN[OHN['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNOREC
ASE)]
OHN2= OHN[OHN['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNOR
ECASE)]
OHN3 = OHN[OHN['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORE
CASE)]
OHN4 = OHN[OHN['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNO
RECASE)]
NegativeN = len(OHN1)+len(OHN2)
PositiveN = len(OHN3)+len(OHN4)
df['User_Location'] = df.User_Location.fillna('')
OH = df[df['User_Location'].str.contains("Ohio|OH", flags=re.IGNORECASE)]
OH1= OH[OH['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNORECASE
)]
OH2= OH[OH['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNORECA
SE)]
OH3 = OH[OH['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORECAS
E)]
OH4 = OH[OH['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNOREC
ASE)]
Negative = len(OH1)+len(OH2)
Positive = len(OH3)+len(OH4)
Only_Neutral = OHN[OHN['sentimentLabel'].str.contains("Neutral", flags=re.IGNO
RECASE)]
Neutral_Final = len(Only_Neutral)
Positive_Final = Positive + PositiveN+Neutral_Final
Negative_Final = Negative + NegativeN
if Negative_Final > Positive_Final:
    Biden_vote_count = Biden_vote_count+16
    print("Biden Wins Ohio")
    print(f"Biden Vote Count: {Biden_vote_count}")
    print('-'*50)
    trump_win.append(0)
    biden_win.append(1)
else:
    Trump_vote_count = Trump_vote_count+16
    print("Trump Wins Ohio")
    print(f"Trump Vote Count: {Trump_vote_count}")
    print('-'*50)
    trump_win.append(1)
    biden_win.append(0)
#MICHIGAN---------------------------------------------------------------------
```

```
----------------------------------------
Neutral['User_Location'] = df.User_Location.fillna("")
MIN = Neutral[Neutral['User_Location'].str.contains("Michigan|MI", flags=re.IG
NORECASE)]
MIN1= MIN[MIN['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNOREC
ASE)]
MIN2= MIN[MIN['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNOR
ECASE)]
MIN3 = MIN[MIN['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORE
CASE)]
MIN4 = MIN[MIN['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNO
RECASE)]
NegativeN = len(MIN1)+len(MIN2)
PositiveN = len(MIN3)+len(MIN4)
df['User_Location'] = df.User_Location.fillna('')
MI = df[df['User_Location'].str.contains("Michigan|MI", flags=re.IGNORECASE)]
MI1= MI[MI['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNORECASE
)]
MI2= MI[MI['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNORECA
SE)]
MI3 = MI[MI['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORECAS
E)]
MI4 = MI[MI['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNOREC
ASE)]
Negative = len(MI1)+len(MI2)
Positive = len(MI3)+len(MI4)
Only_Neutral = MIN[MIN['sentimentLabel'].str.contains("Neutral", flags=re.IGNO
RECASE)]
Neutral_Final = len(Only_Neutral)
import numpy as np
random = np.random.randint(low = 1, high = 10)
if (random%2) == 0:
        Negative_Final = Negative + NegativeN + Neutral_Final
        Positive_Final = Positive + PositiveN
else:
    Positive_Final = Positive + PositiveN + Neutral_Final
    Negative_Final = Negative + NegativeN
if Negative_Final > Positive_Final:
    Biden_vote_count = Biden_vote_count+14
    print("Biden Wins Michigan")
    print(f"Biden Vote Count: {Biden_vote_count}")
    print('-'*50)
    trump_win.append(0)
    biden_win.append(1)
else:
    Trump_vote_count = Trump_vote_count+14
    print("Trump Wins Michigan")
    print(f"Trump Vote Count: {Trump_vote_count}")
    print('-'*50)
    trump_win.append(1)
    biden_win.append(0)
#GEORGIA----------------------------------------------------------------------
----------------------------------------
Neutral['User_Location'] = df.User_Location.fillna("")
GAN = Neutral[Neutral['User_Location'].str.contains("Georgia|GA", flags=re.IGN
ORECASE)]
GAN1= GAN[GAN['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNOREC
```

```
ASE)]
GAN2= GAN[GAN['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNOR
ECASE)]
GAN3 = GAN[GAN['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORE
CASE)]
GAN4 = GAN[GAN['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNO
RECASE)]
NegativeN = len(GAN1)+len(GAN2)
PositiveN = len(GAN3)+len(GAN4)
df['User_Location'] = df.User_Location.fillna('')
GA = df[df['User_Location'].str.contains("Georgia|GA", flags=re.IGNORECASE)]
GA1= GA[GA['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNORECASE
)]
GA2= GA[GA['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNORECA
SE)]
GA3 = GA[GA['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORECAS
E)]
GA4 = GA[GA['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNOREC
ASE)]
Negative = len(GA1)+len(GA2)
Positive = len(GA3)+len(GA4)
Only_Neutral = GAN[GAN['sentimentLabel'].str.contains("Neutral", flags=re.IGNO
RECASE)]
Neutral_Final = len(Only_Neutral)
Positive_Final = Positive + PositiveN+Neutral_Final
Negative_Final = Negative + NegativeN

if Negative_Final > Positive_Final:
    Biden_vote_count = Biden_vote_count+14
    print("Biden Wins Georgia")
    print(f"Biden Vote Count: {Biden_vote_count}")
    print('-'*50)
    trump_win.append(0)
    biden_win.append(1)

else:
    Trump_vote_count = Trump_vote_count+14
    print("Trump Wins Georgia")
    print(f"Trump Vote Count: {Trump_vote_count}")
    print('-'*50)
    trump_win.append(1)
    biden_win.append(0)
#North Carolina-----------------------------------------------------------------
----------------------------------------
Neutral['User_Location'] = df.User_Location.fillna("")
NCN = Neutral[Neutral['User_Location'].str.contains("North Carolina|NC", flags
=re.IGNORECASE)]
NCN1= NCN[NCN['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNOREC
ASE)]
NCN2= NCN[NCN['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNOR
ECASE)]
NCN3 = NCN[NCN['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORE
CASE)]
NCN4 = NCN[NCN['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNO
RECASE)]
NegativeN = len(NCN1)+len(NCN2)
PositiveN = len(NCN3)+len(NCN4)
```

```
df['User_Location'] = df.User_Location.fillna('')
NC = df[df['User_Location'].str.contains("North Carolina|NC", flags=re.IGNOREC
ASE)]
NC1= NC[NC['sentimentLabel'].str.contains("Mild-Negative", flags=re.IGNORECASE
)]
NC2= NC[NC['sentimentLabel'].str.contains("Strong-Negative", flags=re.IGNORECA
SE)]
NC3 = NC[NC['sentimentLabel'].str.contains("Mild-Positive", flags=re.IGNORECAS
E)]
NC4 = NC[NC['sentimentLabel'].str.contains("Strong-Positive", flags=re.IGNOREC
ASE)]
Negative = len(NC1)+len(NC2)
Positive = len(NC3)+len(NC4)
Only_Neutral = NCN[NCN['sentimentLabel'].str.contains("Neutral", flags=re.IGNO
RECASE)]
Neutral_Final = len(Only_Neutral)
import numpy as np
random = np.random.randint(low = 1, high = 10)
if (random%2) == 0:
        Negative_Final = Negative + NegativeN + Neutral_Final
        Positive_Final = Positive + PositiveN
else:
    Positive_Final = Positive + PositiveN + Neutral_Final
    Negative_Final = Negative + NegativeN
if Negative_Final > Positive_Final:
    Biden_vote_count = Biden_vote_count+13
    print("Biden Wins North Carolina")
    print(f"Biden Vote Count: {Biden_vote_count}")
    print('-'*50)
    trump_win.append(0)
    biden_win.append(1)
else:
    Trump_vote_count = Trump_vote_count+13
    print("Trump Wins North Carolina")
    print(f"Trump Vote Count: {Trump_vote_count}")
    print('-'*50)
    trump_win.append(1)
    biden_win.append(0)

#-----------------------------------------------------------------------------
-----------------------------------
if Trump_vote_count > Biden_vote_count:
    print("  *****Trump wins the 2020 election.*****")
else:
    print("  *****Biden wins the 2020 election*****")
print('-'*50)
print(f"Trump's final score was {Trump_vote_count}")
print(f"Biden's final score was {Biden_vote_count}")
print('-'*50)
```

```
/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:6: SettingWithCo
pyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:40: SettingWithC
opyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:49: SettingWithC
opyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:77: SettingWithC
opyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:112: SettingWith
CopyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy

Biden Wins California
Biden Vote Count: 53
--------------------------------------------------
Trump Wins Texas
Trump Vote Count: 36
--------------------------------------------------
Biden Wins New York
Biden Vote Count: 80
--------------------------------------------------
Trump Wins Florida
Trump Vote Count: 63
--------------------------------------------------
```

```
/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:153: SettingWith
CopyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:194: SettingWith
CopyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:228: SettingWith
CopyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:263: SettingWith
CopyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy

Biden Wins Pennsylvania
Biden Vote Count: 98
--------------------------------------------------
Biden Wins Illinois
Biden Vote Count: 116
--------------------------------------------------
Trump Wins Ohio
Trump Vote Count: 79
--------------------------------------------------
Biden Wins Michigan
Biden Vote Count: 130
--------------------------------------------------
Trump Wins Georgia
Trump Vote Count: 93
--------------------------------------------------
Trump Wins North Carolina
Trump Vote Count: 106
--------------------------------------------------
  *****Biden wins the 2020 election*****
--------------------------------------------------
Trump's final score was 106
Biden's final score was 130
--------------------------------------------------
```

```
/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:304: SettingWith
CopyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:341: SettingWith
CopyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
```

# Final DataFrame Of The Results

In [196]:
```
topStates = pd.read_csv("Group_2_Phase_4_Top10States.csv") # The states with t
he most Electoral votes
topStates
topStates["Trump's States"] = trump_win
topStates["Trump's States"].replace(to_replace = 0, value = "Lost", inplace =
True)
topStates["Trump's States"].replace(to_replace = 1, value = "Won", inplace = T
rue)
topStates["Biden's States"] = biden_win
topStates["Biden's States"].replace(to_replace = 0, value = "Lost", inplace =
True)
topStates["Biden's States"].replace(to_replace = 1, value = "Won", inplace = T
rue)
topStates
```

Out[196]:

| | State | Electoral Votes | Difference in % of Votes | Swing State | 2016 Status | Trump's States | Biden's States |
|---|---|---|---|---|---|---|---|
| **0** | California | 53 | 28.80% | No | Blue | Lost | Won |
| **1** | Texas | 36 | 9.20% | No | Red | Won | Lost |
| **2** | New York | 27 | 21.30% | No | Blue | Lost | Won |
| **3** | Florida | 27 | 1.30% | Yes | Red | Won | Lost |
| **4** | Pennsylvania | 18 | 1.20% | Yes | Red | Lost | Won |
| **5** | Illinois | 18 | 16% | No | Blue | Lost | Won |
| **6** | Ohio | 16 | 8.60% | No | Red | Won | Lost |
| **7** | Michigan | 14 | 0.30% | Yes | Red | Lost | Won |
| **8** | Georgia | 14 | 5.70% | No | Red | Won | Lost |
| **9** | North Carolina | 13 | 3.80% | Yes | Red | Won | Lost |

# Geo-Map

In [197]:
```
!pip install geopy
import geopy
from geopy.geocoders import Nominatim
from geopy.geocoders import Nominatim
from mpl_toolkits.basemap import Basemap
nom = Nominatim()
```

```
Requirement already satisfied: geopy in /usr/local/lib/python3.6/site-package
s (1.21.0)
Requirement already satisfied: geographiclib<2,>=1.49 in /usr/local/lib/pytho
n3.6/site-packages (from geopy) (1.50)

/usr/local/lib/python3.6/site-packages/ipykernel_launcher.py:6: DeprecationWa
rning: Using Nominatim with the default "geopy/1.21.0" `user_agent` is strong
ly discouraged, as it violates Nominatim's ToS https://operations.osmfoundati
on.org/policies/nominatim/ and may possibly cause 403 and 429 HTTP errors. Pl
ease specify a custom `user_agent` with `Nominatim(user_agent="my-applicatio
n")` or by overriding the default `user_agent`: `geopy.geocoders.options.defa
ult_user_agent = "my-application"`. In geopy 2.0 this will become an exceptio
n.
```

In [198]:
```
location = df["User_Location"].value_counts()
new_loc = location.to_string()
location #Taking the frequency of each location
locc = pd.DataFrame(location)
locc
e = df.User_Location.dropna()
ee = pd.DataFrame(e)
zg = ee.User_Location.value_counts().head(300)
ZG = zg.index
ZLG= pd.DataFrame(ZG)
ZLG.columns = ["Location"]
ZLG
e = df.User_Location.dropna()
ee = pd.DataFrame(e)
zg = ee.User_Location.value_counts().head(300)
count = pd.DataFrame(zg)
count
ZLG["Coordinates"] = ZLG["Location"].apply(nom.geocode)
ZLG.dropna()
```

Out[198]:

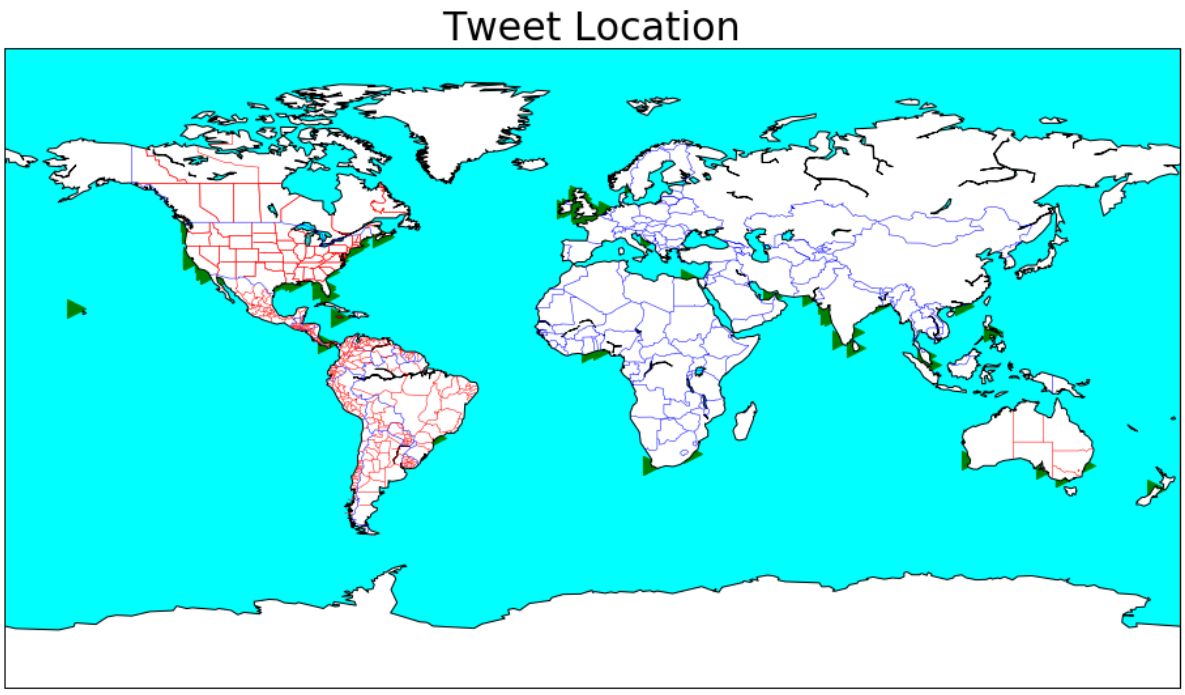
| | Location | Coordinates |
|---|---|---|
| **1** | United States | (United States, (39.7837304, -100.4458825)) |
| **2** | USA | (United States, (39.7837304, -100.4458825)) |
| **3** | California, USA | (California, United States of America, (36.701... |
| **4** | Texas, USA | (Texas, United States of America, (31.8160381,... |
| **5** | India | (भारत - India, (22.3511148, 78.6677428)) |
| **...** | ... | ... |
| **295** | Islamabad, Pakistan | (اسلام آباد, Abbottābād District, وفاقی دارالح... |
| **296** | Amsterdam, The Netherlands | (Amsterdam, Noord-Holland, Nederland, (52.3727... |
| **297** | Bay Area, CA | (San Francisco Bay Area, San Francisco, San Fr... |
| **298** | Montréal, Québec | (Montréal, Agglomération de Montréal, Montréal... |
| **299** | Connecticut | (Connecticut, United States of America, (41.65... |

297 rows × 2 columns

In [199]:
```
ZLG["Latitude"] = ZLG["Coordinates"].apply(lambda x: x.latitude if x != None e
lse None)
ZLG["Longitude"] = ZLG["Coordinates"].apply(lambda x: x.longitude if x != None
else None)
Lat = ZLG["Latitude"]
Long = ZLG["Longitude"]
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns #Statistical Data Visualization
import matplotlib.pyplot as plt
%matplotlib inline
```

In [200]:
```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(20,9))
m = Basemap(projection='gall',
            resolution = 'c')
m.drawcoastlines()
m.drawcountries(color= 'blue')
m.drawstates(color= 'red')
m.drawmapboundary(fill_color='aqua')
m.fillcontinents(color='white',lake_color='aqua')
plt.title("Tweet Location",fontsize = 30)
Lat = list(ZLG["Latitude"])
Long = list(ZLG["Longitude"])
m.scatter(Long,Lat,latlon=True,color="green",marker = ">",s = 200)
plt.show()
```

```
/usr/local/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:4788:
RuntimeWarning: invalid value encountered in greater
  lonsin = np.where(lonsin > lon_0+180, lonsin-360 ,lonsin)
/usr/local/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:4789:
RuntimeWarning: invalid value encountered in less
  lonsin = np.where(lonsin < lon_0-180, lonsin+360 ,lonsin)
/usr/local/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:4795:
RuntimeWarning: invalid value encountered in greater_equal
  itemindex = len(lonsin)-np.where(londiff>=thresh)[0]
/usr/local/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:4826:
RuntimeWarning: invalid value encountered in less
  mask = np.logical_or(lonsin<lon_0-180,lonsin>lon_0+180)
/usr/local/lib/python3.6/site-packages/mpl_toolkits/basemap/__init__.py:4826:
RuntimeWarning: invalid value encountered in greater
  mask = np.logical_or(lonsin<lon_0-180,lonsin>lon_0+180)
```



# Discussions

Starting off with this project, we ran into quite a few roadblocks. We needed the amount of followers a user had for our tweet rating metric and the user location to test our second and third hypotheses. After spending almost the entire day on Thursday trying to find ways to pull user info from our tweets and joining it to our original DataFrame, we had to scratch the idea and move in a different direction. However, we were still very interested in our first hypothesis regarding the number of tweets and Trumps' approval rating - so we went ahead and worked on this part without followers or location. We made visualizations with two y-axes to compare the approval ratings by week (our third party data) and spent most of the day Friday creating new DataFrames that broke down specific keywords, polarity scores, or a combination of the two. We were then able to produce visualizations to test our remaining hypotheses - which will be discussed below.

Another issue we ran into had to do with the number of tweets we originally scraped - we scraped for 2000 instances of our keyword per week for 17 weeks with 9 keywords, which gave us about 300,000 tweets in our final DataFrame. Working with a DataFrame of this size in our virtual containers proved to be extremely inefective, so we figured the best course of action would be to eliminate some keywords and scrape less tweets per week to ensure we still got the same distribution of tweets per week.

# Conclusions

Let's explore how each hypothesis turned out

1) President Trump’s approval rating has increased with the number of “positive sentiment” tweets about the stimulus checks distributed by the US government. The more tweets involving stimulus checks that have highly rated words as determined by sentiment analysis, the higher we believe Trump’s approval rating will be. Then see how these results relate in our election simulation.

- Given that the TextBlob tool pooled most of our tweets into the 'neutral' polarity category, it is tough to pinpoint exactly if positive or negative sentiment tweets specifically had a correlation to Trump's approval rating. However, when we looked at tweets broadly, it seems as if the more tweets there were about stimulus checks, coronavirus, and tweets we scraped for in general, the lower Trump's approval rating was. They seemed to move almost inversely - it's likely that the tweets were not a direct cause of the rating, but they were just correlated. We did some research to find out some interesting occurences on the timeline of Trump's coronavirus actions and added them to our plots as well to show that there may have been other outside factors at play.
- Despite our issues with the sentiment tool, we were able to see a bit of a direct correlation between the number of negative tweets about stimulus checks and Trump's approval rating. This can be seen in figure 3.5.4 - the red line (negative tweets) and green line (approval rating) seem to generally move together. The same cannot be said, however, about the number of negative tweets about coronavirus (shown in section 3.5.5) as there appears to be much less correlation. However, neutral tweets still do generally move inversely with approval rating in this figure.
- Given the results from the simulation, we conclude that President Trump should turn his attention to winning swing states, becuase he is still winning his historically Republican states, but not the swing states. This may be an attainable task for him, becuase as we saw in Part 1, it appears that his approval rating is on the rise.

2) Sentiment ratings for the tweets will show an increase after the stimulus checks actually start to be distributed.

- We were able to explore this via sections 3.5.3, 3.5.4, and 3.5.5. We were not able to see a clear decrease in negative sentiment tweets after the stimulus checks were announced, or when they started to be distributed. We would have expected to see a sharp drop in our red line (number of negative tweets) after around week 12 or 13, but this did not occur on any of our plots.