In [1]:
```
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np
import re
import nltk
import preprocessor as p

import matplotlib.pyplot as plt
import plotly.express as ex
import plotly.graph_objs as go
import plotly.offline as pyo
from plotly.subplots import make_subplots
plt.rc('figure',figsize=(17,13))


nltk.download('vader_lexicon')
from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA
from wordcloud import WordCloud,STOPWORDS

from pandas.plotting import autocorrelation_plot
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.seasonal import seasonal_decompose
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/chrissymo/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

In [2]:
```
df = pd.read_csv('/Users/chrissymo/Documents/MSIS/research/with Vivian/COVID-19/vaccine tweets/fetch data/complete dataset for vaccine_DecToApr.csv
```

```
/Users/chrissymo/opt/anaconda3/lib/python3.8/site-packages/IPython/core/interactiveshell.py:3146: DtypeWarning:

Columns (12) have mixed types.Specify dtype option on import or set low_memory=False.
```

In [94]:
```
--version
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-94-4bc6a30f992c> in <module>
----> 1 --version

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

In [4]:
```
df = df.drop_duplicates(subset=['text'])
```

In [5]:
```
len(df)
```

Out[5]: 169061

In [6]:
```
df1 = df[['date','text']]
```

In [7]:
```
df1['text'][1]
```

Out[7]: 'Understanding the Different Types of #PPE for Workplace Safety https://t.co/twqd3sRdgL (https://t.co/twqd3sRdgL) #health #covidvaccine'

In [8]:
```
f_data = df1

f_data.text =f_data.text.str.lower()

#Remove twitter handlers
f_data.text = f_data.text.apply(lambda x:re.sub('@[^\s]+','',x))

#remove hashtags
#f_data.text = f_data.text.apply(lambda x:re.sub(r'\B#\S+','',x))

# Remove URLS
f_data.text = f_data.text.apply(lambda x:re.sub(r"http\S+", "", x))

# Remove all the special characters
f_data.text = f_data.text.apply(lambda x:' '.join(re.findall(r'\w+', x)))

#remove all single characters
f_data.text = f_data.text.apply(lambda x:re.sub(r'\s+[a-zA-Z]\s+', '', x))

# Substituting multiple spaces with single space
f_data.text = f_data.text.apply(lambda x:re.sub(r'\s+', ' ', x, flags=re.I))


#remove HTML spcial entities (e.g. &amp)
f_data.text = f_data.text.apply(lambda x:re.sub(r'\&\w*;','',x))

#remove words with 2 or fewer letters
f_data.text = f_data.text.apply(lambda x:re.sub(r'\b\w{1,2}\b','',x))
```

```
/Users/chrissymo/opt/anaconda3/lib/python3.8/site-packages/pandas/core/generic.py:5494: SettingWithCopyWarning:


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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

In [9]:
```
f_data['text'][1]
```

Out[9]:
```
'understanding the different types  ppe for workplace safety health covidvaccine'
```

In [10]:
```
f_data = f_data.drop_duplicates(subset=['text'])
```

In [11]:
```
len(f_data)
```

Out[11]:
```
158366
```

In [12]:
```
sid = SIA()
f_data['sentiments']           = f_data['text'].apply(lambda x: sid.polarity_scores(' '.join(re.findall(r'\w+',x.lower()))))
f_data['Positive Sentiment']   = f_data['sentiments'].apply(lambda x: x['pos'])
f_data['Neutral Sentiment']    = f_data['sentiments'].apply(lambda x: x['neu'])
f_data['Negative Sentiment']   = f_data['sentiments'].apply(lambda x: x['neg'])
f_data['Compound']             = f_data['sentiments'].apply(lambda x: x['compound'])
f_data.drop(columns=['sentiments'],inplace=True)
```

```
<ipython-input-12-7196e9d8a2c0>:2: SettingWithCopyWarning:


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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

<ipython-input-12-7196e9d8a2c0>:3: SettingWithCopyWarning:


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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

<ipython-input-12-7196e9d8a2c0>:4: SettingWithCopyWarning:


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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

<ipython-input-12-7196e9d8a2c0>:5: SettingWithCopyWarning:


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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

<ipython-input-12-7196e9d8a2c0>:6: SettingWithCopyWarning:


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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

/Users/chrissymo/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py:4308: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

In [13]:
```
f_data
```

Out[13]:

	date	text	Positive Sentiment	Neutral Sentiment	Negative Sentiment	Compound
0	12/8/20	coronavirus cases around the world have now re...	0.073	0.725	0.202	-0.5423
1	12/8/20	understanding the different types ppe for wor...	0.237	0.763	0.000	0.4215
2	12/8/20	the tuesday said the the lead agency the co...	0.000	1.000	0.000	0.0000
3	12/8/20	covidvaccine found delhi mask social dista...	0.000	1.000	0.000	0.0000
4	12/8/20	pre orders covid vaccine top five billion co...	0.184	0.816	0.000	0.2023
...	...	...	...	...	...	...
169074	4/8/21	vaccinateeveryindian but possible also add so...	0.000	1.000	0.000	0.0000
169075	4/8/21	when will you recognize those who received ...	0.000	1.000	0.000	0.0000
169076	4/8/21	maharashtra news school education department m...	0.206	0.794	0.000	0.3818
169077	4/8/21	the two nurses who administered covid19 vaccin...	0.000	1.000	0.000	0.0000
169078	4/8/21	allocate 3000 crore ramp the production ...	0.197	0.803	0.000	0.4019

158366 rows × 6 columns

In [14]:
```
f_data['date'] = pd.to_datetime(f_data['date']).dt.date
```

```
<ipython-input-14-491773b2ee27>:1: SettingWithCopyWarning:
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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

In [2]:
```
## could read clean sentiment score from csv file
#f_data.to_csv('/Users/chrissymo/Documents/MSIS/research/with Vivian/COVID-19/vaccine tweets/DATA FOR PAPER/DectoApr_sentiment_6_4_v2.csv')
f_data = pd.read_csv('/Users/chrissymo/Documents/MSIS/research/with Vivian/COVID-19/vaccine tweets/DATA FOR PAPER/DectoApr_sentiment_6_4_v2.csv')
```

In [108]:
```
df_mean
```

Out[108]:

	date	Unnamed: 0	Unnamed: 0.1	Positive Sentiment	Neutral Sentiment	Negative Sentiment	Compound
0	2020-12-08	243.0	260.121150	0.115257	0.848552	0.034152	0.152305
1	2020-12-09	534.5	589.072917	0.106615	0.861375	0.032031	0.120250
2	2020-12-12	612.0	682.576271	0.114695	0.845780	0.039525	0.150203
3	2020-12-13	688.5	763.755319	0.096064	0.865426	0.038500	0.116862
4	2020-12-14	803.0	881.644444	0.098548	0.877726	0.023733	0.153564
...	...	...	...	...	...	...	...
115	2021-04-04	154843.0	165276.671772	0.096628	0.855586	0.047775	0.079305
116	2021-04-05	155348.0	165817.524412	0.095461	0.866072	0.038467	0.100313
117	2021-04-06	156015.0	166538.247119	0.102136	0.855117	0.042753	0.100639
118	2021-04-07	157140.5	167736.938095	0.095839	0.856242	0.047926	0.085282
119	2021-04-08	158120.5	168809.500000	0.076165	0.869790	0.054049	0.035850

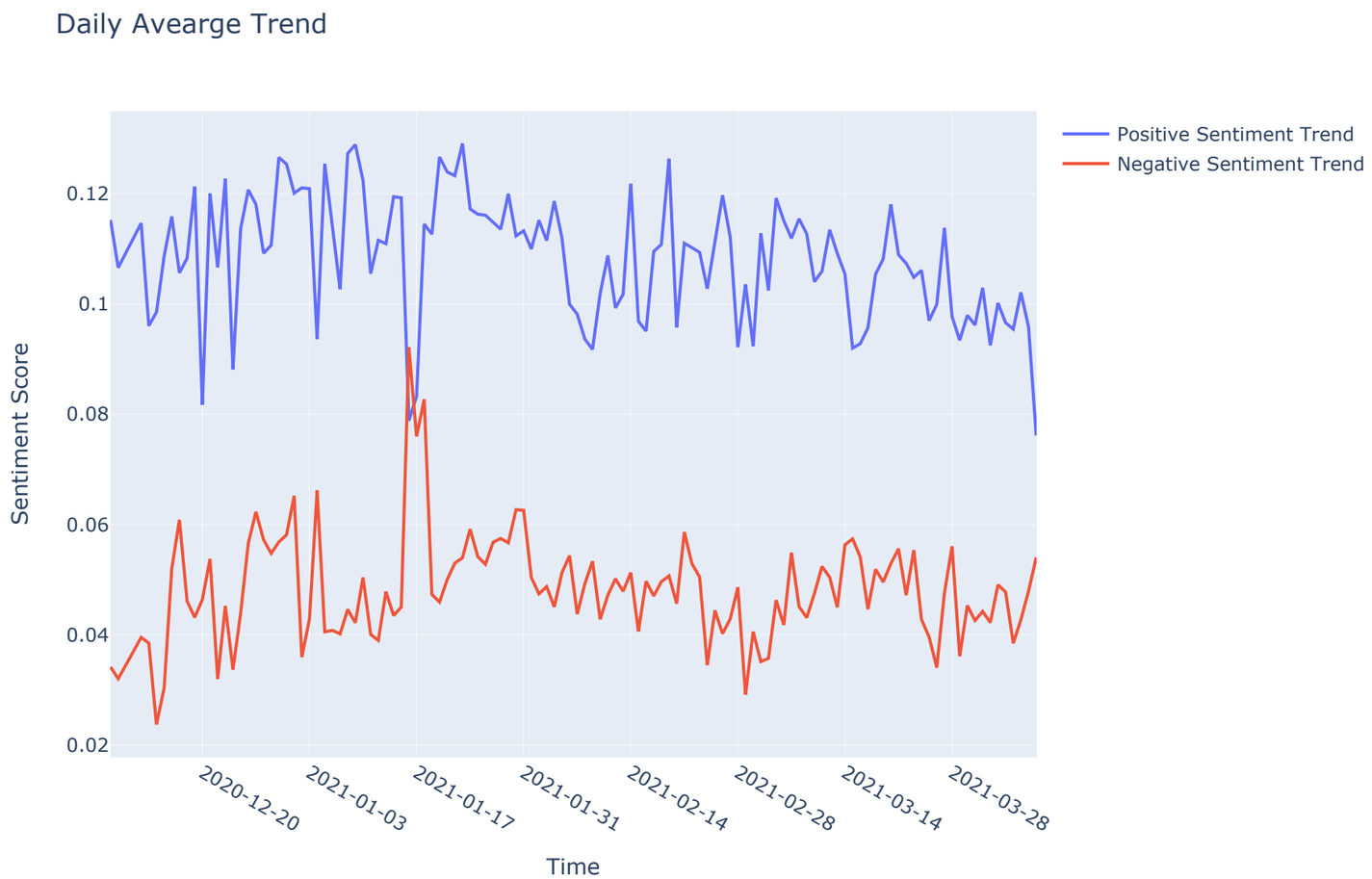
120 rows × 7 columns

In [120]:
```
df_mean = f_data.groupby(by='date').mean().reset_index()

fig = make_subplots(rows=1, cols=1)

fig.add_trace(go.Scatter(x=df_mean['date'],y=df_mean['Positive Sentiment'],name='Positive Sentiment Trend'),
    row=1, col=1)
fig.add_trace(go.Scatter(x=df_mean['date'],y=df_mean['Negative Sentiment'],name='Negative Sentiment Trend'),
    row=1, col=1)

fig.update_layout(height=600, width=900, title_text="Daily Avearge Trend",xaxis_title="Time",yaxis_title="Sentiment Score")
fig.update_xaxes(
    ticktext=["2020-12-20", "2021-01-03", "2021-01-17", "2021-01-31","2021-02-14","2021-02-28","2021-03-14","2021-03-28"],
    tickvals=["2020-12-20", "2021-01-03", "2021-01-17", "2021-01-31","2021-02-14","2021-02-28","2021-03-14","2021-03-28"],
)
fig.show()
```



In [ ]:

In [ ]:
```
#f_data = pd.read_csv('DectoApr_sentiment.csv')
```

In [19]:
```
polarity = []
for i in range(len(f_data)):
    if f_data['Compound'][i] > 0.001:
        if f_data['Positive Sentiment'][i] > 0.5:
            re = 'Highly Positive'
        else:
            re = 'Positive'
    elif f_data['Compound'][i] < -0.001:
        if f_data['Negative Sentiment'][i] > 0.5:
            re = 'Highly Negative'
        else:
            re = 'Negative'
    else:
        re = 'Neutral'
    polarity.append(re)
```

In [20]:
```
f_data['Polarity'] = polarity
```

In [24]:
```
f_data.to_csv('/Users/chrissymo/Documents/MSIS/research/with Vivian/COVID-19/vaccine tweets/DATA FOR PAPER/test_6_4/DectoApr_sentiment_5categories_(
```

In [4]:
```
#f_data = pd.read_csv('DectoApr_sentiment_5categories_1.csv')
f_data = pd.read_csv('/Users/chrissymo/Documents/MSIS/research/with Vivian/COVID-19/vaccine tweets/DATA FOR PAPER/test_6_4/DectoApr_sentiment_5categ
```

In [5]:
```
f_data = f_data.drop_duplicates(subset=['text'])
```

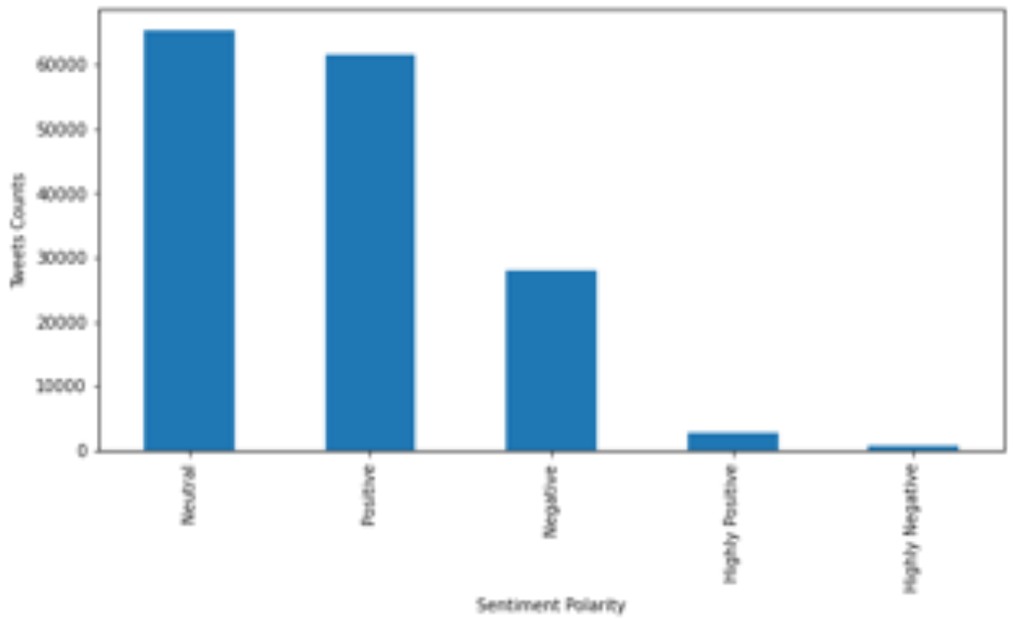
In [6]:
```
len(f_data)
```

Out[6]:
```
158366
```

In [7]:
```
f_data['Polarity'].value_counts()
```

Out[7]:
```
Neutral            65345
Positive           61598
Negative           27944
Highly Positive     2858
Highly Negative      621
Name: Polarity, dtype: int64
```

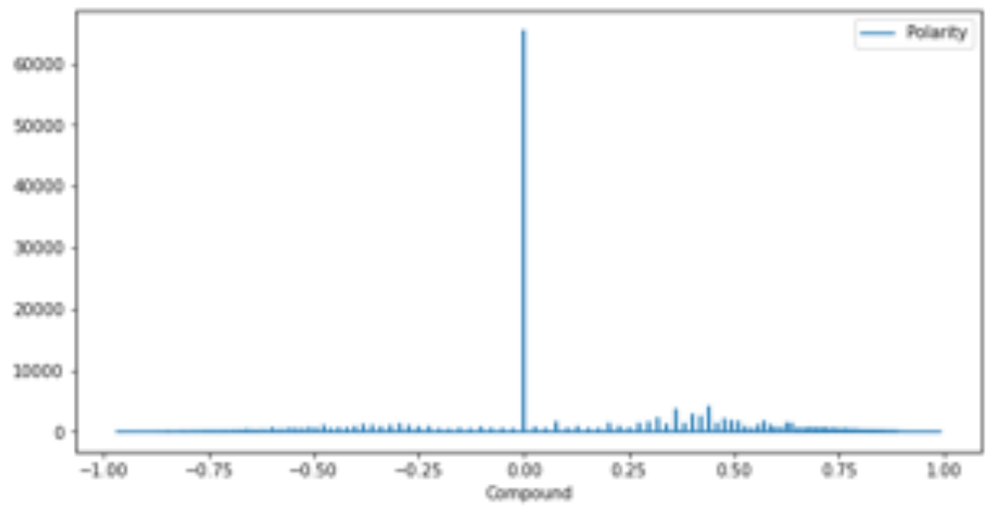
In [15]:
```
f_data['Polarity'].value_counts().to_frame().plot(kind='bar',figsize = (10,5),legend = False)
plt.ylabel('Tweets Counts')
plt.xlabel('Sentiment Polarity')
plt.show()
```



In [25]:
```
df_compound = f_data.groupby(by='Compound').count()
```

In [26]:
```
counts = df_compound['Polarity']
counts.to_frame().plot(figsize = (10,5))
```

Out[26]: <AxesSubplot:xlabel='Compound'>



In [8]:
```
pos = f_data[f_data['Polarity'] == 'Positive']
hpos = f_data[f_data['Polarity'] == 'Highly Positive']
neu = f_data[f_data['Polarity'] == 'Neutral']
neg = f_data[f_data['Polarity'] == 'Negative']
hneg = f_data[f_data['Polarity'] == 'Highly Negative']
```

In [9]:
```
pos_all = pd.concat([pos,hpos]).sort_values(by='date')
neg_all = pd.concat([neg,hneg]).sort_values(by='date')
```

In [10]:
```
pos_mean = pos_all.groupby(by='date').mean().reset_index()
neg_mean = neg_all.groupby(by='date').mean().reset_index()
```

In [30]:
```
fig = make_subplots(rows=2, cols=1)

fig.add_trace(go.Scatter(x=pos_mean['date'],y=pos_mean['Compound'],name='POS Compound Trend'),
    row=1, col=1)

#fig.add_annotation(x="2020-09-08", y=0.875001,
            #text="The most Positive Sentiment:2020-09-08",
            #showarrow=True,
            #arrowhead=1)

fig.add_trace(go.Scatter(x=neg_mean['date'],y=neg_mean['Compound'],name='Neg Compound Trend'),
    row=2, col=1)

#fig.add_annotation(x="2020-09-30", y=-0.7173365,
            #text="The most Negative Sentiment:2020-09-30",
            #showarrow=True,
            #arrowhead=1,row=2, col=1)

fig.update_layout(height=600, width=900, title_text="Daily Avearge Trend")
fig.show()
```

...

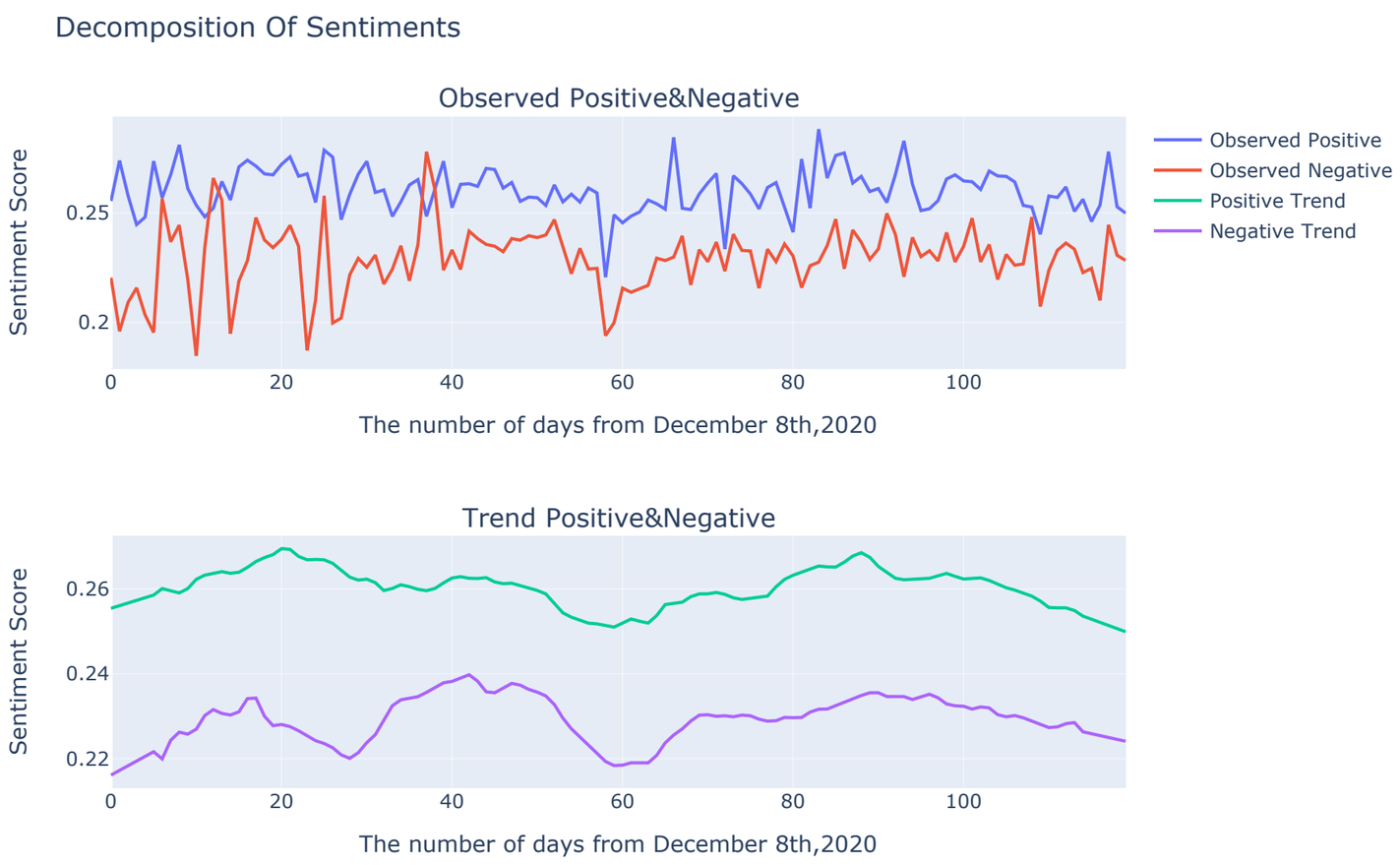
In [95]:
```
fig = make_subplots(rows=2, cols=1, subplot_titles=('Observed Positive&Negative','Trend Positive&Negative'))

res_pos = seasonal_decompose(pos_mean['Positive Sentiment'], period=12, model='additive', extrapolate_trend='freq')
res_neg = seasonal_decompose(neg_mean['Negative Sentiment'], period=12, model='additive', extrapolate_trend='freq')

fig.add_trace(go.Scatter(x=np.arange(0,len(res_pos.observed)), y=res_pos.observed,name='Observed Positive'),
    row=1, col=1)
fig.add_trace(go.Scatter(x=np.arange(0,len(res_neg.observed)), y=res_neg.observed,name='Observed Negative'),
    row=1, col=1)


fig.add_trace(go.Scatter(x=np.arange(0,len(res_pos.trend)), y=res_pos.trend,name='Positive Trend'),
    row=2, col=1)
fig.add_trace(go.Scatter(x=np.arange(0,len(res_neg.trend)), y=res_neg.trend,name='Negative Trend'),
    row=2, col=1)

fig.update_layout(height=600, width=900, title_text="Decomposition Of Sentiments")
fig.update_yaxes(title_text="Sentiment Score")
fig.update_xaxes(title_text="The number of days from December 8th,2020")
fig.show()
```



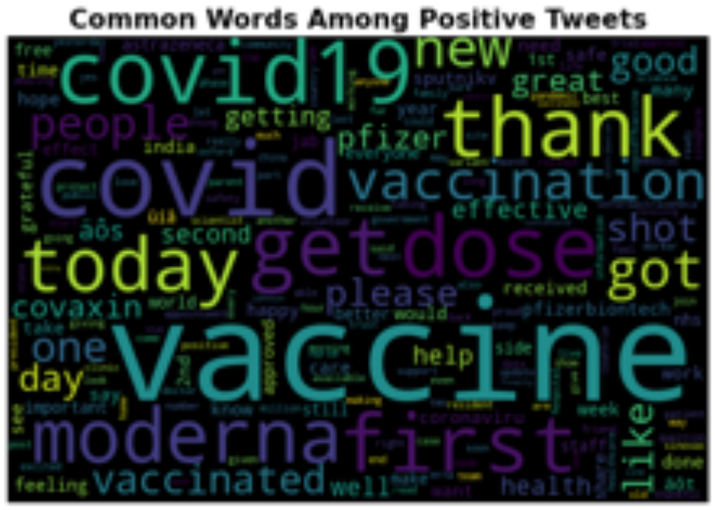
In [32]:
```
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
stop_words.extend(['amp','covidvaccine'])
#stop_words.extend(['amp','vaccine','covid','vaccines'])'Äôs'
```

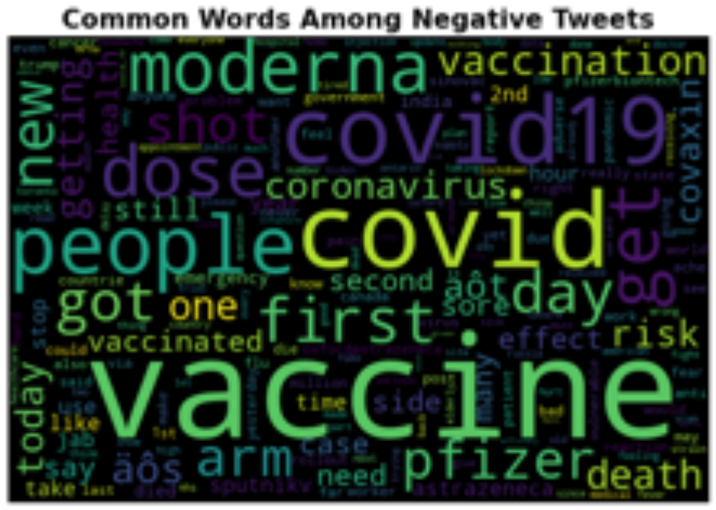
In [33]:
```
Positive_text = ' '.join(pos.text)
Negative_text = ' '.join(neg.text)


pwc = WordCloud(stopwords=stop_words,width=600,height=400,collocations = False).generate(Positive_text)
nwc = WordCloud(stopwords=stop_words,width=600,height=400,collocations = False).generate(Negative_text)

plt.subplot(1,2,1)
plt.title('Common Words Among Positive Tweets',fontsize=16,fontweight='bold')
plt.imshow(pwc)
plt.axis('off')
plt.subplot(1,2,2)
plt.title('Common Words Among Negative Tweets',fontsize=16,fontweight='bold')
plt.imshow(nwc)
plt.axis('off')

plt.show()
```



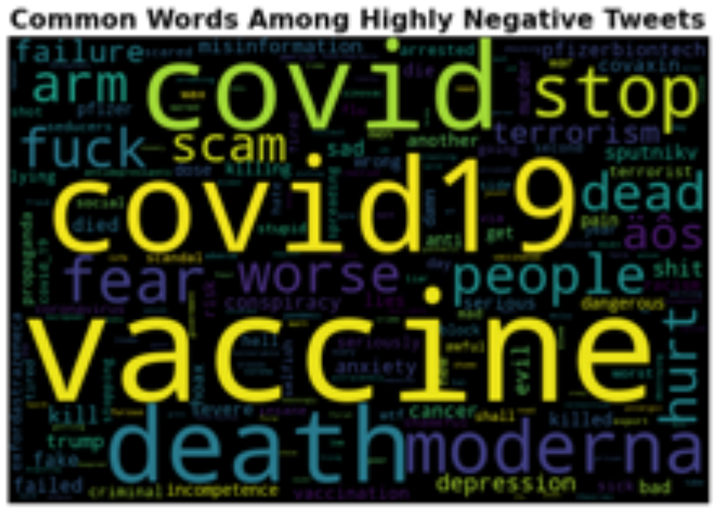


In [34]:
```
Positive_text = ' '.join(hpos.text)
Negative_text = ' '.join(hneg.text)


pwc = WordCloud(stopwords=stop_words,width=600,height=400,collocations = False).generate(Positive_text)
nwc = WordCloud(stopwords=stop_words,width=600,height=400,collocations = False).generate(Negative_text)

plt.subplot(1,2,1)
plt.title('Common Words Among Highly Positive Tweets',fontsize=16,fontweight='bold')
plt.imshow(pwc)
plt.axis('off')
plt.subplot(1,2,2)
plt.title('Common Words Among Highly Negative Tweets',fontsize=16,fontweight='bold')
plt.imshow(nwc)
plt.axis('off')

plt.show()
```



In [35]:
```
## volume of pos,neg,neu, percentage
f_data['month']        = pd.DatetimeIndex(f_data['date']).month
```

In [36]:
```
len(f_data)
```

Out[36]:
```
158366
```

In [37]:
```
f_data['month'].value_counts()
```

Out[37]:
```
1     60840
3     38702
2     36226
12    16492
4      6106
Name: month, dtype: int64
```

In [38]:
```
f_12 = f_data[f_data['month'] == 12]
f_1 = f_data[f_data['month'] == 1]
f_2 = f_data[f_data['month'] == 2]
f_3 = f_data[f_data['month'] == 3]
f_4 = f_data[f_data['month'] == 4]
```

In [39]:
```
df_month = pd.DataFrame(columns = f_12['Polarity'].value_counts().index.tolist(),data = [f_12['Polarity'].value_counts().values,
                                                                                          f_1['Polarity'].value_counts().values,
                                                                                          f_2['Polarity'].value_counts().values,
                                                                                          f_3['Polarity'].value_counts().values,
                                                                                          f_4['Polarity'].value_counts().values,],
                                                                                  index = ['December','January','February','March','April'])
```

In [40]:
```
df_month
```

Out[40]:

	Positive	Neutral	Negative	Highly Positive	Highly Negative
December	6475	6400	3197	351	69
January	24795	23534	11103	1132	276
February	15089	14097	6310	606	124
March	17411	14160	6308	685	138
April	2911	2071	1026	84	14

In [ ]:
```

```

topic modelling

In [41]:
```
import re
import numpy as np
import pandas as pd
from pprint import pprint

# Gensim
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel

# spacy for lemmatization
import spacy

# Plotting tools
import pyLDAvis
import pyLDAvis.gensim  # don't skip this
import matplotlib.pyplot as plt
%matplotlib inline

# Enable logging for gensim - optional
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.ERROR)

import warnings
warnings.filterwarnings("ignore",category=DeprecationWarning)
```

In [42]:
```
from nltk.corpus import stopwords
stop_words_tm = stopwords.words('english')
#stop_words_tm.extend(['amp'])
stop_words_tm.extend(['amp','covidvaccine','covid','vaccine', 'vaccines', 'vaccination','vaccinate'])
```

In [43]:
```

#data = pos_all.text
data = neu.text
#data = neg_all.text
```

In [44]:
```
len(data)
```

Out[44]: 65345

In [142]:
```
neg_all['week'] = pd.DatetimeIndex(neg_all['date']).week
```

```
<ipython-input-142-c8ee33fddd0f>:1: FutureWarning:

weekofyear and week have been deprecated, please use DatetimeIndex.isocalendar().week instead, which returns a Series.  To exactly reproduce the behavior of w
ek and weekofyear and return an Index, you may call pd.Int64Index(idx.isocalendar().week)
```

In [45]:
```
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))  # deacc=True removes punctuations

data_words = list(sent_to_words(data))

# Build the bigram and trigram models
bigram = gensim.models.Phrases(data_words, min_count=5, threshold=100) # higher threshold fewer phrases.
trigram = gensim.models.Phrases(bigram[data_words], threshold=100)

# Faster way to get a sentence clubbed as a trigram/bigram
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)

# Define functions for stopwords, bigrams, trigrams and lemmatization
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if word not in stop_words_tm] for doc in texts]

def make_bigrams(texts):
    return [bigram_mod[doc] for doc in texts]

def make_trigrams(texts):
    return [trigram_mod[bigram_mod[doc]] for doc in texts]

def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    """https://spacy.io/api/annotation"""
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])
    return texts_out

# Remove Stop Words
data_words_nostops = remove_stopwords(data_words)

# Form Bigrams
data_words_bigrams = make_bigrams(data_words_nostops)

# Initialize spacy 'en' model, keeping only tagger component (for efficiency)
# python3 -m spacy download en
nlp = spacy.load('en', disable=['parser', 'ner'])

# Do lemmatization keeping only noun, adj, vb, adv
data_lemmatized = lemmatization(data_words_bigrams, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV'])

# Create Dictionary
id2word = corpora.Dictionary(data_lemmatized)

# Create Corpus
texts = data_lemmatized

# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]

# View
print(corpus[:1])
```

```
[[(0, 1)]]
```

In [123]:
```
##look for the optimal topic number function

def compute_coherence_values(dictionary, corpus, texts, limit, start=2, step=3):
    coherence_values = []
    model_list = []
    for num_topics in range(start, limit, step):
        model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                                num_topics=num_topics,
                                                id2word=dictionary,
                                                random_state=100,
                                                update_every=1,
                                                chunksize=100,
                                                passes=10,
                                                alpha='auto',
                                                per_word_topics=True)
        model_list.append(model)
        coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=dictionary, coherence='c_v')
        coherence_values.append(coherencemodel.get_coherence())

    return model_list, coherence_values
```

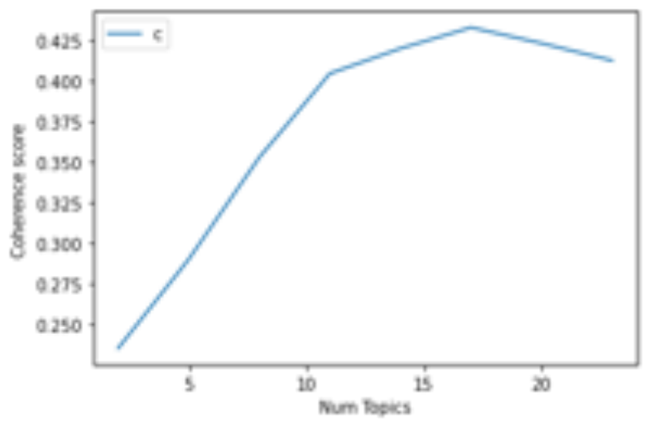
negative topics

In [53]:
```
model_list, coherence_values = compute_coherence_values(dictionary=id2word, corpus=corpus, texts=data_lemmatized, start=2, limit=26, step=3)
```

In [51]:
```
# DO NOT COVER THIS PART
# remove key terms
limit=26; start=2; step=3;
x = range(start, limit, step)
plt.plot(x, coherence_values)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```

...

In [54]:
```
# remove key terms
limit=26; start=2; step=3;
x = range(start, limit, step)
plt.plot(x, coherence_values)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```



In [55]:
```
# DO NOT COVER THIS PART
# Print the coherence scores
for m, cv in zip(x, coherence_values):
    print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 2  has Coherence Value of 0.2351
Num Topics = 5  has Coherence Value of 0.29
Num Topics = 8  has Coherence Value of 0.3528
Num Topics = 11  has Coherence Value of 0.4044
Num Topics = 14  has Coherence Value of 0.4198
Num Topics = 17  has Coherence Value of 0.4327
Num Topics = 20  has Coherence Value of 0.423
Num Topics = 23  has Coherence Value of 0.4124
```

In [ ]:
```

```

In [242]:
```
# Build LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=17,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha='auto',
                                            per_word_topics=True)
```

In [243]:
```
# Compute Perplexity
print('\nPerplexity: ', lda_model.log_perplexity(corpus))  # a measure of how good the model is. lower the better.

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model, texts=data_lemmatized, dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Perplexity:  -19.65419410362789

Coherence Score:  0.43274900186822207
```

In [244]:
```
pprint(lda_model.print_topics())
```

```
...
```

In [245]:
```
# Visualize the topics
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
vis
```

```
...
```

In [246]:
```
def format_topics_sentences(ldamodel=None, corpus=corpus, texts=data):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row_list in enumerate(ldamodel[corpus]):
        row = row_list[0] if ldamodel.per_word_topics else row_list
        # print(row)
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each document
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df = sent_topics_df.append(pd.Series([int(topic_num), round(prop_topic,4), topic_keywords]), ignore_index=True)
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords']

    # Add original text to the end of the output
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)


df_topic_sents_keywords = format_topics_sentences(ldamodel=lda_model, corpus=corpus, texts=data_lemmatized)
```

In [247]:
```
df_topic_sents_keywords_neg = df_topic_sents_keywords
```

In [248]:
```
# Number of Documents for Each Topic
topic_counts = df_topic_sents_keywords_neg['Dominant_Topic'].value_counts()

# Percentage of Documents for Each Topic
topic_contribution = round(topic_counts/topic_counts.sum(), 4)
```

In [249]:
```
dict_keyword_prob = {}
for i in range(0,17):
    wp = lda_model.show_topic(i)
    dict_keyword_prob[i] = wp
```

In [250]:
```
dict_keywords = {}
for i in range(0,17):
    wp = lda_model.show_topic(i)
    topic_keywords = ", ".join([word for word, prop in wp])

    dict_keywords[i] = topic_keywords
```

In [251]:
```
# FOR TEST
neg_topic_dis = pd.DataFrame({"Topic_ counts":topic_counts,'percentage_contribution':topic_contribution},index = topic_counts.index)
neg_topic_dis = neg_topic_dis.reset_index()
neg_topic_dis = neg_topic_dis.rename(columns={'index':'Topic'})
neg_topic_dis['keywords'] = neg_topic_dis['Topic'].apply(lambda x : dict_keywords[x])
neg_topic_dis['keyword_prob'] = neg_topic_dis['Topic'].apply(lambda x : dict_keyword_prob[x])
neg_topic_dis
```

Out[251]:

	Topic	Topic_ counts	percentage_contribution	keywords	keyword_prob
0	11.0	10526	0.3685	dose, arm, today, day, second, shoot, sore, fe...	[(dose, 0.12979399), (arm, 0.10206086), (today...
1	5.0	3428	0.1200	get, bad, force, government, due, even, never,...	[(get, 0.4135061), (bad, 0.10761349), (force, ...
2	8.0	2804	0.0982	still, death, yesterday, may, far, report, fig...	[(still, 0.12206295), (death, 0.107017115), (y...
3	2.0	2695	0.0943	die, stop, new, many, reporting_case, problem,...	[(die, 0.12941256), (stop, 0.108661555), (new,...
4	6.0	1812	0.0634	take, hour, work, back, refuse, show, next, hi...	[(take, 0.24330258), (hour, 0.109030135), (wor...
5	1.0	1667	0.0584	say, make, know, come, last, authorize, yet, w...	[(say, 0.20955278), (make, 0.15449382), (know,...
6	0.0	1320	0.0462	people, risk, time, kill, around, young, beat,...	[(people, 0.37958357), (risk, 0.15238975), (ti...
7	16.0	1207	0.0423	want, need, find, world, continue, question, l...	[(want, 0.110355966), (need, 0.10272887), (fin...
8	7.0	864	0.0302	first, receive, case, news, approve, fake, wai...	[(first, 0.25113449), (receive, 0.119474694), ...
9	9.0	729	0.0255	use, go, send, would, part, thing, hear, trump...	[(use, 0.23565203), (go, 0.1649664), (send, 0....
10	3.0	694	0.0243	could, read, week, lose, try, ever, put, worry...	[(could, 0.0964104), (read, 0.07566973), (week...
11	15.0	439	0.0154	country, poor, already, order, announce, compa...	[(country, 0.21528456), (poor, 0.090330176), (...
12	13.0	304	0.0106	year, side, hard, big, let, effort, remember, ...	[(year, 0.10952472), (side, 0.06690505), (hard...
13	10.0	28	0.0010	virus, hurt, finally, allow, suck, corona, tru...	[(virus, 0.16185042), (hurt, 0.103654295), (fi...
14	12.0	22	0.0008	much, lockdown, doubt, pay, completely, reach,...	[(much, 0.13006245), (lockdown, 0.07760127), (...
15	4.0	14	0.0005	warn, whole, drug, critical, vac, else, reveal...	[(warn, 0.10009732), (whole, 0.052548867), (dr...
16	14.0	12	0.0004	call, seem, pfizerbiontech, site, vial, race, ...	[(call, 0.15114096), (seem, 0.11017945), (pfiz...

In [ ]:
```
```

In [265]:
```
neg_topic_dis.to_csv('/Users/chrissymo/Documents/MSIS/research/with Vivian/COVID-19/vaccine tweets/DATA FOR PAPER/test_6_4/neg_topic_distribution_3
```

In [252]:
```
neg_all = neg_all.reset_index().drop(columns = 'index')
```

In [253]:
```
df_topic_sents_keywords_neg['week'] = neg_all['week']
```

In [254]:
```
df_topic_sents_keywords_neg.week.unique().tolist()
```

Out[254]:
```
[50, 51, 52, 53, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

In [255]:
```
list_week = df_topic_sents_keywords.week.unique().tolist()
```

In [256]:
```
frames = []
for i in list_week:
    df = df_topic_sents_keywords_neg[df_topic_sents_keywords_neg['week'] == i]['Dominant_Topic'].value_counts().reset_index().sort_values(['index']
    df = df.rename(columns={'Dominant_Topic': 'week'+str(i)})
    frames.append(df)
```

In [257]:
```
neg_disbyweek = pd.concat([frames[0],frames[1],frames[2],frames[3],frames[4],frames[5],frames[6],frames[7],frames[8],
                           frames[9],frames[10],frames[11],frames[12],frames[13],frames[14],frames[15],frames[16],frames[17]],axis=1)
```

In [258]:
```
neg_disbyweek
```

Out[258]:

index	week50	week51	week52	week53	week1	week2	week3	week4	week5	week6	week7	week8	week9	week10	week11	week12	week13	week14
0.0	3	7.0	35.0	110.0	46.0	140	165.0	237.0	118.0	12.0	75	87.0	53.0	114.0	38.0	18.0	34.0	28.0
1.0	5	8.0	46.0	170.0	55.0	116	155.0	305.0	127.0	26.0	100	102.0	69.0	164.0	91.0	35.0	48.0	45.0
2.0	12	8.0	77.0	277.0	106.0	241	237.0	492.0	206.0	54.0	173	149.0	92.0	247.0	131.0	78.0	77.0	38.0
3.0	3	5.0	23.0	65.0	25.0	58	60.0	132.0	80.0	10.0	39	32.0	32.0	54.0	34.0	22.0	12.0	8.0
4.0	3	NaN	NaN	2.0	NaN	2	NaN	1.0	2.0	NaN	1	NaN	2.0	NaN	NaN	NaN	NaN	1.0
5.0	7	15.0	105.0	311.0	119.0	299	329.0	568.0	303.0	51.0	202	220.0	148.0	333.0	126.0	99.0	112.0	81.0
6.0	10	12.0	45.0	179.0	69.0	158	175.0	293.0	142.0	46.0	127	115.0	101.0	146.0	81.0	27.0	44.0	42.0
7.0	4	10.0	25.0	89.0	29.0	89	68.0	136.0	69.0	12.0	52	55.0	36.0	75.0	46.0	16.0	31.0	22.0
8.0	6	7.0	60.0	269.0	97.0	229	298.0	442.0	240.0	52.0	184	176.0	126.0	299.0	133.0	57.0	79.0	50.0
9.0	4	5.0	24.0	86.0	23.0	52	62.0	145.0	60.0	9.0	41	37.0	27.0	74.0	31.0	10.0	21.0	18.0
10.0	3	NaN	2.0	6.0	2.0	2	4.0	1.0	1.0	1.0	1	2.0	NaN	2.0	NaN	NaN	1.0	NaN
11.0	13	45.0	228.0	989.0	410.0	909	950.0	1663.0	816.0	231.0	669	661.0	462.0	1074.0	449.0	334.0	417.0	206.0
12.0	3	NaN	1.0	6.0	NaN	1	2.0	5.0	NaN	NaN	1	2.0	1.0	NaN	NaN	NaN	NaN	NaN
13.0	3	1.0	9.0	43.0	10.0	22	34.0	51.0	30.0	2.0	15	18.0	9.0	37.0	5.0	5.0	5.0	5.0
14.0	2	NaN	NaN	NaN	2.0	2	NaN	NaN	2.0	1.0	1	1.0	NaN	1.0	NaN	NaN	NaN	NaN
15.0	7	1.0	9.0	33.0	13.0	33	33.0	74.0	31.0	9.0	22	32.0	23.0	47.0	37.0	12.0	14.0	9.0
16.0	11	7.0	34.0	107.0	40.0	102	107.0	225.0	126.0	25.0	73	75.0	44.0	114.0	45.0	26.0	26.0	20.0

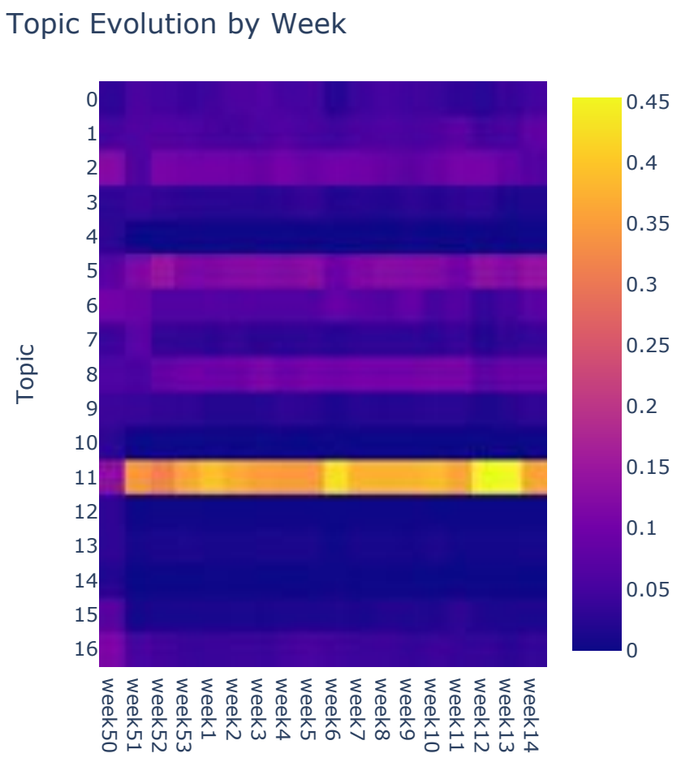
In [259]:
```
neg_disbyweek.to_csv('/Users/chrissymo/Documents/MSIS/research/with Vivian/COVID-19/vaccine tweets/DATA FOR PAPER/test_6_4/neg_dis_byweek_v3.csv')
```

In [87]:
```
week = pd.read_excel('/Users/chrissymo/Documents/MSIS/research/with Vivian/COVID-19/vaccine tweets/DATA FOR PAPER/test_6_4/neg_dis_byweek_v4.xlsx')
```

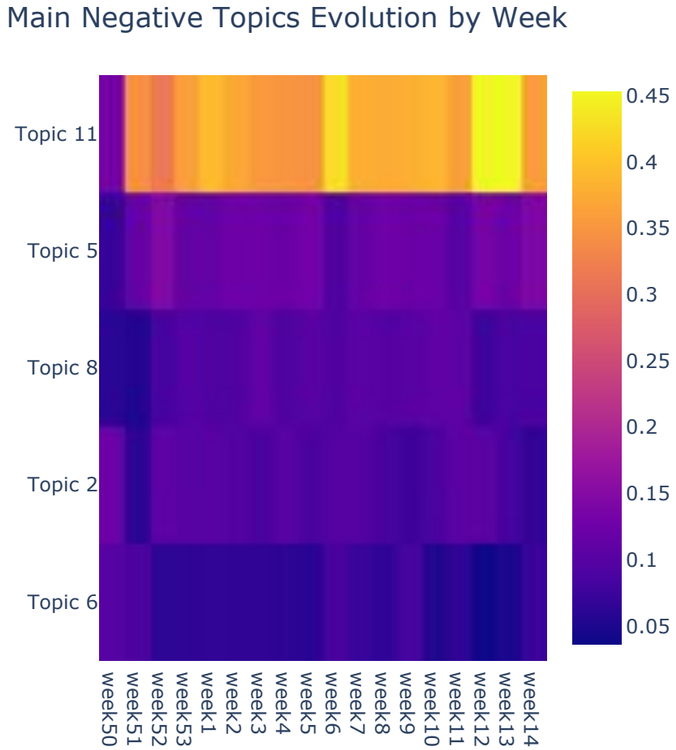
In [33]:
```
neg_dis_byweek
```

...

In [35]:
```
# DO NOT COVER THIS PART
dis = neg_dis_byweek.set_index('Topic')
import plotly.express as px
fig = px.imshow(dis)
fig.update_layout(height=500, width=450,title='Topic Evolution by Week')
fig.update_yaxes(dtick=1)
fig.show()
```



In [92]:
```
#dis = neg_dis_byweek.set_index('Topic')
dis = neg_dis_byweek.iloc[:,1:]
import plotly.express as px
fig = px.imshow(dis, y=['Topic 11','Topic 5','Topic 8','Topic 2','Topic 6'],aspect='auto')
fig.update_layout(height=500, width=450,title='Main Negative Topics Evolution by Week')
#fig.update_yaxes(ticktext=[2,5,6,8,11])
fig.show()
```



In [ ]:

In [ ]:

In [54]:
```
# DO NOT COVER THIS PART
neg_topic_dis = pd.DataFrame({"Topic_ counts":topic_counts,'percentage_contribution':topic_contribution},index = topic_counts.index)
neg_topic_dis = neg_topic_dis.reset_index()
neg_topic_dis = neg_topic_dis.rename(columns={'index':'Topic'})
neg_topic_dis['keywords'] = neg_topic_dis['Topic'].apply(lambda x : dict_keywords[x])
neg_topic_dis['keyword_prob'] = neg_topic_dis['Topic'].apply(lambda x : dict_keyword_prob[x])
neg_topic_dis
```

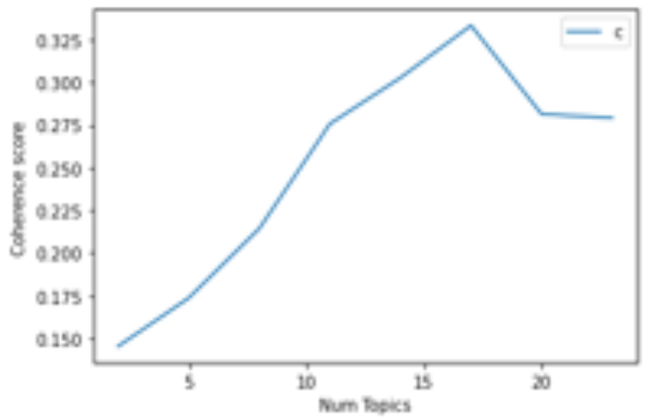
Out[54]:

	Topic	Topic_ counts	percentage_contribution	keywords	keyword_prob
0	9.0	30050	0.9287	vaccine, covid, covaxin, vaccination, country,...	[(vaccine, 0.44583222), (covid, 0.20571028), (...
1	12.0	2128	0.0658	get, arm, second, sore, shoot, feel, side_effe...	[(get, 0.21611035), (arm, 0.08328044), (second...
2	4.0	115	0.0036	dose, first, go, use, still, well, come, probl...	[(dose, 0.17618996), (first, 0.14231405), (go,...
3	15.0	32	0.0010	hour, give, new, hurt, think, really, call, wo...	[(hour, 0.1276519), (give, 0.10295799), (new, ...
4	1.0	14	0.0004	covidvaccine, death, far, test, absolutely, po...	[(covidvaccine, 0.29290953), (death, 0.1501720...
5	13.0	4	0.0001	make, bad, government, even, much, poor, final...	[(make, 0.21638979), (bad, 0.18473224), (gover...
6	10.0	3	0.0001	risk, want, send, part, soon, continue, medium...	[(risk, 0.17480937), (want, 0.15904461), (send...
7	11.0	2	0.0001	need, virus, last, back, show, begin, safe, do...	[(need, 0.12803362), (virus, 0.1005116), (last...
8	2.0	2	0.0001	time, do, let, read, symptom, never, lose, har...	[(time, 0.16881865), (do, 0.08870502), (let, 0...
9	3.0	2	0.0001	would, tell, trial, pay, thing, lockdown, mark...	[(would, 0.14425857), (tell, 0.13257223), (tri...
10	14.0	1	0.0000	people, find, believe, food, social_distancing...	[(people, 0.5618986), (find, 0.111821204), (be...
11	7.0	1	0.0000	day, today, many, next, long, worry, critical,...	[(day, 0.29470232), (today, 0.281723), (many, ...
12	16.0	1	0.0000	take, approve, refuse, price, rate, mean, nati...	[(take, 0.37846065), (approve, 0.09017692), (r...
13	6.0	1	0.0000	russian, fail, kill, high, datum, young, ignor...	[(russian, 0.14076237), (fail, 0.1171826), (ki...
14	0.0	1	0.0000	vaccinate, week, work, s, doctor, affect, rese...	[(vaccinate, 0.18852848), (week, 0.17079894), ...

positive

In [87]:
```
model_list, coherence_values = compute_coherence_values(dictionary=id2word, corpus=corpus, texts=data_lemmatized, start=2, limit=26, step=3)
```

In [88]:
```
# FOR TEST
limit=26; start=2; step=3;
x = range(start, limit, step)
plt.plot(x, coherence_values)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```



In [89]:
```
# Print the coherence scores
for m, cv in zip(x, coherence_values):
    print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 2  has Coherence Value of 0.1454
Num Topics = 5  has Coherence Value of 0.1737
Num Topics = 8  has Coherence Value of 0.2144
Num Topics = 11  has Coherence Value of 0.2757
Num Topics = 14  has Coherence Value of 0.3027
Num Topics = 17  has Coherence Value of 0.3333
Num Topics = 20  has Coherence Value of 0.2814
Num Topics = 23  has Coherence Value of 0.2792
```

In [49]:
```
# DO NOT COVER THIS PART
limit=26; start=2; step=3;
x = range(start, limit, step)
plt.plot(x, coherence_values)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```

...

In [50]:
```
# Print the coherence scores
for m, cv in zip(x, coherence_values):
    print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

...

In [135]:
```
# Build LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=17,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=500,
                                            passes=10,
                                            alpha='auto',
                                            per_word_topics=True)
```

In [136]:
```
# Compute Perplexity
print('\nPerplexity: ', lda_model.log_perplexity(corpus))  # a measure of how good the model is. lower the better.

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model, texts=data_lemmatized, dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Perplexity:  -14.362453919077458

Coherence Score:  0.3282426559622823
```

In [137]:
```
# Visualize the topics
pyLDAvis.enable_notebook()
#vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word, mds='mmds')
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
vis
```

. . .

In [138]:
```
# FOR TEST

def format_topics_sentences(ldamodel=None, corpus=corpus, texts=data):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row_list in enumerate(ldamodel[corpus]):
        row = row_list[0] if ldamodel.per_word_topics else row_list
        # print(row)
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each document
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df = sent_topics_df.append(pd.Series([int(topic_num), round(prop_topic,4), topic_keywords]), ignore_index=True)
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords']

    # Add original text to the end of the output
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)


df_topic_sents_keywords = format_topics_sentences(ldamodel=lda_model, corpus=corpus, texts=data_lemmatized)

# Number of Documents for Each Topic
topic_counts = df_topic_sents_keywords['Dominant_Topic'].value_counts()

# Percentage of Documents for Each Topic
topic_contribution = round(topic_counts/topic_counts.sum(), 4)

dict_keyword_prob = {}
for i in range(0,17):
    wp = lda_model.show_topic(i)
    dict_keyword_prob[i] = wp


dict_keywords = {}
for i in range(0,17):
    wp = lda_model.show_topic(i)
    topic_keywords = ", ".join([word for word, prop in wp])

    dict_keywords[i] = topic_keywords


pos_topic_dis = pd.DataFrame({"Topic_ counts":topic_counts,'percentage_contribution':topic_contribution},index = topic_counts.index)
pos_topic_dis = pos_topic_dis.reset_index()
pos_topic_dis = pos_topic_dis.rename(columns={'index':'Topic'})
pos_topic_dis['keywords'] = pos_topic_dis['Topic'].apply(lambda x : dict_keywords[x])
pos_topic_dis['keyword_prob'] = pos_topic_dis['Topic'].apply(lambda x : dict_keyword_prob[x])
pos_topic_dis
```

Out[138]:

	Topic	Topic_ counts	percentage_contribution	keywords	keyword_prob
0	8.0	24912	0.3865	get, today, feel, receive, shot, do, health, r...	[(get, 0.31696063), (today, 0.14694697), (feel...

	Topic	Topic_ counts	percentage_contribution	keywords	keyword_prob
1	3.0	7335	0.1138	first, dose, grateful, happy, would, super, pr...	[(first, 0.25413945), (dose, 0.2516258), (grat...
2	1.0	7055	0.1095	thank, work, far, new, part, scientist, report...	[(thank, 0.34075454), (work, 0.058828484), (fa...
3	14.0	5543	0.0860	take, safe, protect, free, hope, easy, datum, ...	[(take, 0.17713484), (safe, 0.120698914), (pro...
4	16.0	4171	0.0647	second, well, go, number, start, year, finally...	[(second, 0.1431294), (well, 0.13854288), (go,...
5	5.0	3746	0.0581	approve, vaccinated, help, use, want, country,...	[(approve, 0.11760163), (vaccinated, 0.1070519...
6	10.0	3687	0.0572	good, yesterday, share, fully, moderna, side_e...	[(good, 0.20743261), (yesterday, 0.068291456),...
7	7.0	2596	0.0403	covaxin, say, give, love, trial, fine, end, fo...	[(covaxin, 0.14604595), (say, 0.1454159), (giv...
8	15.0	837	0.0130	people, many, government, wait, important, eff...	[(people, 0.2008341), (many, 0.06287175), (gov...
9	6.0	708	0.0110	vaccinate, make, also, even, available, update...	[(vaccinate, 0.18817455), (make, 0.13137275), ...
10	11.0	705	0.0109	need, soon, support, could, tell, call, parent...	[(need, 0.085046865), (soon, 0.05718621), (sup...
11	2.0	704	0.0109	see, friend, news, family, positive, bring, te...	[(see, 0.11161567), (friend, 0.0747361), (news...
12	4.0	541	0.0084	great, pfizerbiontech, science, approval, than...	[(great, 0.13583447), (pfizerbiontech, 0.08836...
13	12.0	515	0.0080	effective, morning, read, ask, back, must, hig...	[(effective, 0.16941135), (morning, 0.06693147...
14	0.0	496	0.0077	show, still, come, really, find, already, incr...	[(show, 0.07977074), (still, 0.07552852), (com...
15	13.0	476	0.0074	day, know, world, amazing, let, live, true, ma...	[(day, 0.20200372), (know, 0.08394105), (world...
16	9.0	429	0.0067	time, doctor, staff, vaccination, thing, check...	[(time, 0.11284623), (doctor, 0.05213245), (st...

In [139]:
```
pos_topic_dis.to_csv('/Users/chrissymo/Documents/MSIS/research/with Vivian/COVID-19/vaccine tweets/DATA FOR PAPER/test_6_4/pos_topic_distribution_3
```

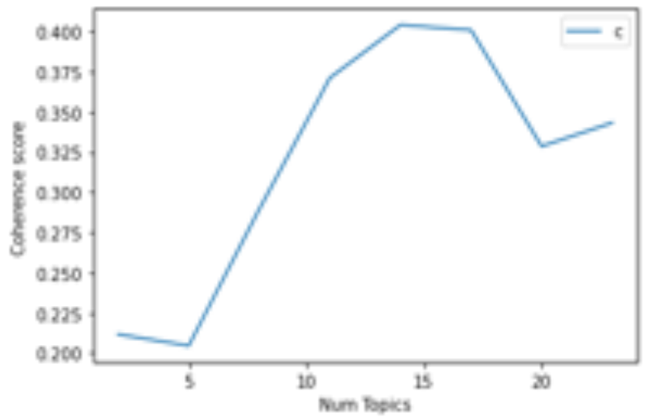
In [ ]:
```
```

neutral

In [ ]:
```
```

In [124]:
```
model_list, coherence_values = compute_coherence_values(dictionary=id2word, corpus=corpus, texts=data_lemmatized, start=2, limit=26, step=3)
```

In [98]:
```
limit=26; start=2; step=3;
x = range(start, limit, step)
plt.plot(x, coherence_values)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```



In [99]:
```
# Print the coherence scores
for m, cv in zip(x, coherence_values):
    print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 2  has Coherence Value of 0.2117
Num Topics = 5  has Coherence Value of 0.2046
Num Topics = 8  has Coherence Value of 0.2896
Num Topics = 11  has Coherence Value of 0.3711
Num Topics = 14  has Coherence Value of 0.404
Num Topics = 17  has Coherence Value of 0.4008
Num Topics = 20  has Coherence Value of 0.3286
Num Topics = 23  has Coherence Value of 0.3432
```

In [167]:
```
limit=26; start=2; step=3;
x = range(start, limit, step)
plt.plot(x, coherence_values)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```

...

In [168]:
```
# Print the coherence scores
for m, cv in zip(x, coherence_values):
    print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

...

In [46]:
```
# Build LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=17,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=1000,
                                            passes=10,
                                            alpha='auto',
                                            per_word_topics=True)
```

In [47]:
```
# Compute Perplexity
print('\nPerplexity: ', lda_model.log_perplexity(corpus))  # a measure of how good the model is. lower the better.

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model, texts=data_lemmatized, dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Perplexity:  -13.52456051700191

Coherence Score:  0.4016861162500209
```

In [48]:
```
# Visualize the topics
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
vis
```

Out[48]:

In [49]:
```
##FOR TEST

def format_topics_sentences(ldamodel=None, corpus=corpus, texts=data):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row_list in enumerate(ldamodel[corpus]):
        row = row_list[0] if ldamodel.per_word_topics else row_list
        # print(row)
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each document
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df = sent_topics_df.append(pd.Series([int(topic_num), round(prop_topic,4), topic_keywords]), ignore_index=True)
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords']

    # Add original text to the end of the output
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)


df_topic_sents_keywords = format_topics_sentences(ldamodel=lda_model, corpus=corpus, texts=data_lemmatized)

# Number of Documents for Each Topic
topic_counts = df_topic_sents_keywords['Dominant_Topic'].value_counts()

# Percentage of Documents for Each Topic
topic_contribution = round(topic_counts/topic_counts.sum(), 4)

dict_keyword_prob = {}
for i in range(0,17):
    wp = lda_model.show_topic(i)
    dict_keyword_prob[i] = wp


dict_keywords = {}
for i in range(0,17):
    wp = lda_model.show_topic(i)
    topic_keywords = ", ".join([word for word, prop in wp])

    dict_keywords[i] = topic_keywords


topic_dis = pd.DataFrame({"Topic_ counts":topic_counts,'percentage_contribution':topic_contribution},index = topic_counts.index)
topic_dis = topic_dis.reset_index()
topic_dis = topic_dis.rename(columns={'index':'Topic'})
topic_dis['keywords'] = topic_dis['Topic'].apply(lambda x : dict_keywords[x])
topic_dis['keyword_prob'] = topic_dis['Topic'].apply(lambda x : dict_keyword_prob[x])
topic_dis
```

Out[49]:

	Topic	Topic_ counts	percentage_contribution	keywords	keyword_prob
0	12.0	46302	0.7086	get, dose, first, today, receive, shot, vaccin...	[(get, 0.20970762), (dose, 0.2092874), (first,...

	Topic	Topic_ counts	percentage_contribution	keywords	keyword_prob
1	5.0	6513	0.0997	say, covaxin, trial, see, come, government, up...	[(say, 0.18094581), (covaxin, 0.15485491), (tr...
2	0.0	5356	0.0820	vaccinate, people, give, fully, many, could, b...	[(vaccinate, 0.20739433), (people, 0.14980282)...
3	16.0	1033	0.0158	new, case, hour, report, side_effect, efficacy...	[(new, 0.09339416), (case, 0.06436839), (hour,...
4	3.0	694	0.0106	shoot, make, arrive, available, tomorrow, stat...	[(shoot, 0.17904022), (make, 0.10498332), (arr...
5	7.0	600	0.0092	take, year, read, hospital, right, vaccination...	[(take, 0.30836186), (year, 0.074875966), (rea...
6	14.0	578	0.0088	second, day, know, complete, study, open, covi...	[(second, 0.26611304), (day, 0.12155806), (kno...
7	13.0	566	0.0087	start, production, line, follow, issue, roll, ...	[(start, 0.124776416), (production, 0.06264163...
8	1.0	519	0.0079	age, would, chinese, morning, soon, next, even...	[(age, 0.063679464), (would, 0.063402094), (ch...
9	15.0	475	0.0073	use, also, test, company, question, schedule, ...	[(use, 0.14011964), (also, 0.077866), (test, 0...
10	10.0	465	0.0071	go, need, appointment, full, post, news, least...	[(go, 0.12193465), (need, 0.10426803), (appoin...
11	6.0	454	0.0069	week, last, look, arm, health, pharmacy, manuf...	[(week, 0.12158798), (last, 0.0828681), (look,...
12	11.0	407	0.0062	country, time, month, produce, develop, base, ...	[(country, 0.12435054), (time, 0.100119665), (...
13	8.0	382	0.0058	moderna, call, offer, talk, become, old, plan,...	[(moderna, 0.09195597), (call, 0.062348362), (...
14	9.0	364	0.0056	do, eligible, work, book, find, late, check, v...	[(do, 0.20634356), (eligible, 0.07456031), (wo...
15	2.0	361	0.0055	begin, wait, still, may, announce, long, price...	[(begin, 0.101219684), (wait, 0.09910541), (st...
16	4.0	276	0.0042	already, russian, child, think, world, supply,...	[(already, 0.067583844), (russian, 0.067387395...

In [50]:
```
topic_dis.to_csv('/Users/chrissymo/Documents/MSIS/research/with Vivian/COVID-19/vaccine tweets/DATA FOR PAPER/test_6_4/neu_topic_distribution_3cate
```

In [293]:
```
##FOR TEST

def format_topics_sentences(ldamodel=None, corpus=corpus, texts=data):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row_list in enumerate(ldamodel[corpus]):
        row = row_list[0] if ldamodel.per_word_topics else row_list
        # print(row)
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each document
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df = sent_topics_df.append(pd.Series([int(topic_num), round(prop_topic,4), topic_keywords]), ignore_index=True)
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords']

    # Add original text to the end of the output
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)


df_topic_sents_keywords = format_topics_sentences(ldamodel=lda_model, corpus=corpus, texts=data_lemmatized)

# Number of Documents for Each Topic
topic_counts = df_topic_sents_keywords['Dominant_Topic'].value_counts()

# Percentage of Documents for Each Topic
topic_contribution = round(topic_counts/topic_counts.sum(), 4)

dict_keyword_prob = {}
for i in range(0,17):
    wp = lda_model.show_topic(i)
    dict_keyword_prob[i] = wp


dict_keywords = {}
for i in range(0,17):
    wp = lda_model.show_topic(i)
    topic_keywords = ", ".join([word for word, prop in wp])

    dict_keywords[i] = topic_keywords


topic_dis = pd.DataFrame({"Topic_ counts":topic_counts,'percentage_contribution':topic_contribution},index = topic_counts.index)
topic_dis = topic_dis.reset_index()
topic_dis = topic_dis.rename(columns={'index':'Topic'})
topic_dis['keywords'] = topic_dis['Topic'].apply(lambda x : dict_keywords[x])
topic_dis['keyword_prob'] = topic_dis['Topic'].apply(lambda x : dict_keyword_prob[x])
topic_dis
```

...

In [ ]:

In [70]:
```
##DO NOT COVER THIS PART

def format_topics_sentences(ldamodel=None, corpus=corpus, texts=data):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row_list in enumerate(ldamodel[corpus]):
        row = row_list[0] if ldamodel.per_word_topics else row_list
        # print(row)
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each document
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df = sent_topics_df.append(pd.Series([int(topic_num), round(prop_topic,4), topic_keywords]), ignore_index=True)
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords']

    # Add original text to the end of the output
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)


df_topic_sents_keywords = format_topics_sentences(ldamodel=lda_model, corpus=corpus, texts=data_lemmatized)

# Number of Documents for Each Topic
topic_counts = df_topic_sents_keywords['Dominant_Topic'].value_counts()

# Percentage of Documents for Each Topic
topic_contribution = round(topic_counts/topic_counts.sum(), 4)

dict_keyword_prob = {}
for i in range(0,17):
    wp = lda_model.show_topic(i)
    dict_keyword_prob[i] = wp


dict_keywords = {}
for i in range(0,17):
    wp = lda_model.show_topic(i)
    topic_keywords = ", ".join([word for word, prop in wp])

    dict_keywords[i] = topic_keywords


topic_dis = pd.DataFrame({"Topic_ counts":topic_counts,'percentage_contribution':topic_contribution},index = topic_counts.index)
topic_dis = topic_dis.reset_index()
topic_dis = topic_dis.rename(columns={'index':'Topic'})
topic_dis['keywords'] = topic_dis['Topic'].apply(lambda x : dict_keywords[x])
topic_dis['keyword_prob'] = topic_dis['Topic'].apply(lambda x : dict_keyword_prob[x])
topic_dis
```

...

In [ ]:

In [294]:
```
topic_dis.to_csv('/Users/chrissymo/Documents/MSIS/research/with Vivian/COVID-19/vaccine tweets/DATA FOR PAPER/test_6_4/neu_topic_distribution_3cate
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: