In [1]:
```
import pandas as pd
data = pd.read_csv("/Users/gurnoorvirdi/Desktop/NLP- social sciences/NLP Final
```

In [2]:
```
data
```

Out[2]:

	user_name	user_location	user_description	user_created	user_followers	user_frien
0	ᴛᴛi☻ᵢ₵†	astroworld	wednesday addams as a disney princess keepin i...	2017-05-26 05:46:42	624	9
1	Tom Basile 🇺🇸	New York, NY	Husband, Father, Columnist & Commentator. Auth...	2009-04-16 20:06:23	2253	16
2	Time4fisticuffs	Pewee Valley, KY	#Christian #Catholic #Conservative #Reagan #Re...	2009-02-28 18:57:41	9275	95
3	ethel mertz	Stuck in the Middle	#Browns #Indians #ClevelandProud #[]_[] #Cavs ...	2019-03-07 01:45:06	197	9
4	DIPR-J&K	Jammu and Kashmir	🖋Official Twitter handle of Department of Inf...	2017-02-12 06:45:15	101009	1
...	...	...	...	...	...	
179103	AJIMATI AbdulRahman O.	Ilorin, Nigeria	Animal Scientist|| Muslim|| Real Madrid/Chelsea	2013-12-30 18:59:19	412	16
179104	Jason	Ontario	When your cat has more baking soda than Ninja ...	2011-12-21 04:41:30	150	1
179105	BEEHEMOTH ⏳	🇨🇦 Canada	⚒ The Architects of Free Trade ⚒ Really Did ...	2016-07-13 17:21:59	1623	21
179106	Gary DelPonte	New York City	Global UX UI Visual Designer. StoryTeller, Mus...	2009-10-27 17:43:13	1338	1
179107	TUKY II	Aliwal North, South Africa	TOKELO SEKHOPA | TUKY II | LAST BORN | EISH TU...	2018-04-14 17:30:07	97	16

179108 rows × 13 columns

In [3]:
```
import sys
sys.path
sys.executable
sys.path.insert(0, r"/Users/gurnoorvirdi/anaconda3/lib/python3.11/site-packages
```

In [4]:
```
import nltk
from nrclex import NRCLex
import re

text = data['text']

# Word Tokenization
tokens_list = text.apply(nltk.word_tokenize)

# Get rid of symbols and special characters from each tweet
def remove_special_characters(tokens):
    # Regular expression pattern to match any non-alphanumeric character (inclu
    pattern = r'[^a-zA-Z0-9\s]'
    # Remove special characters and symbols using the pattern
    tokens = [re.sub(pattern, '', token) for token in tokens]
    # Remove any empty tokens
    tokens = [token for token in tokens if token]
    return tokens

# Apply the remove_special_characters function to each list of tokens
tokens_list_cleaned = tokens_list.apply(remove_special_characters)
tokens_list_cleaned
```

Out[4]:
```
0         [If, I, smelled, the, scent, of, hand, sanitiz...
1         [Hey, Yankees, YankeesPR, and, MLB, would, nt,...
2         [diane3443, wdunlap, realDonaldTrump, Trump, n...
3         [brookbanktv, The, one, gift, COVID19, has, gi...
4         [25, July, Media, Bulletin, on, Novel, CoronaV...
                                ...
179103    [Thanks, IamOhmai, for, nominating, me, for, t...
179104    [2020, The, year, of, insanity, Lol, COVID19, ...
179105    [CTVNews, A, powerful, painting, by, Juan, Luc...
179106    [More, than, 1200, students, test, positive, f...
179107    [I, stop, when, I, see, a, Stop, SABCNews, Izi...
Name: text, Length: 179108, dtype: object
```

In [5]:
```
# Sentiment Analysis with NRC
from nrclex import NRCLex

def get_sentiment_score(tokens):
    text = " ".join(tokens)
    return NRCLex(text).affect_frequencies

# Apply the get_sentiment_score function to each list of tokens
sentiment_scores = tokens_list_cleaned.apply(get_sentiment_score)
sentiment_scores
```

Out[5]:
```
0         {'fear': 0.0, 'anger': 0.0, 'anticip': 0.0, 't...
1         {'fear': 0.0, 'anger': 0.0, 'anticip': 0.0, 't...
2         {'fear': 0.0, 'anger': 0.16666666666666666, 'a...
3         {'fear': 0.0, 'anger': 0.0, 'anticip': 0.0, 't...
4         {'fear': 0.0, 'anger': 0.0, 'anticip': 0.0, 't...
                                ...
179103    {'fear': 0.3333333333333333, 'anger': 0.333333...
179104    {'fear': 0.2, 'anger': 0.2, 'anticip': 0.0, 't...
179105    {'fear': 0.125, 'anger': 0.125, 'anticip': 0.0...
179106    {'fear': 0.0, 'anger': 0.0, 'anticip': 0.0, 't...
179107    {'fear': 0.0, 'anger': 0.0, 'anticip': 0.0, 't...
Name: text, Length: 179108, dtype: object
```

In [6]:
```
import nltk
from nrclex import NRCLex
import re
import pandas as pd

text = data['text']

# Word Tokenization
tokens_list = text.apply(nltk.word_tokenize)

# Get rid of symbols and special characters from each tweet
def remove_special_characters(tokens):
    pattern = r'[^a-zA-Z0-9\s]'
    tokens = [re.sub(pattern, '', token) for token in tokens]
    tokens = [token for token in tokens if token]
    return tokens

# Apply the remove_special_characters function to each list of tokens
tokens_list_cleaned = tokens_list.apply(remove_special_characters)

# Sentiment Analysis using NRC Lexicon
def get_sentiment_score(tokens):
    text = " ".join(tokens)
    return NRCLex(text).affect_frequencies

# Apply the get_sentiment_score function to each list of tokens
sentiment_scores = tokens_list_cleaned.apply(get_sentiment_score)

# Create a new DataFrame for sentiment analysis results
sentiment_df = pd.DataFrame(sentiment_scores.tolist(), index=data.index)

# Combine the new DataFrame with the original data
data_with_sentiment = pd.concat([data, sentiment_df], axis=1)

# Display the data with sentiment analysis results
print(data_with_sentiment.head())
```

```
         user_name         user_location  \
0           ᴠ̈ᵢ☻ᵢ₵₹            astroworld
1    Tom Basile 🇺🇸           New York, NY
2  Time4fisticuffs      Pewee Valley, KY
3      ethel mertz  Stuck in the Middle
4         DIPR-J&K     Jammu and Kashmir

                                  user_description         user_created  \
0  wednesday addams as a disney princess keepin i...  2017-05-26 05:46:42
1  Husband, Father, Columnist & Commentator. Auth...  2009-04-16 20:06:23
2  #Christian #Catholic #Conservative #Reagan #Re...  2009-02-28 18:57:41
3  #Browns #Indians #ClevelandProud #[]_[] #Cavs ...  2019-03-07 01:45:06
4  🖊Official Twitter handle of Department of Inf...  2017-02-12 06:45:15

   user_followers  user_friends  user_favourites  user_verified  \
0             624           950            18775          False
1            2253          1677               24           True
2            9275          9525             7254          False
3             197           987             1488          False
4          101009           168              101          False

                 date                                                text  \
0  2020-07-25 12:27:21  If I smelled the scent of hand sanitizers toda...
1  2020-07-25 12:27:17  Hey @Yankees @YankeesPR and @MLB - wouldn't it...
2  2020-07-25 12:27:14  @diane3443 @wdunlap @realDonaldTrump Trump nev...
3  2020-07-25 12:27:10  @brookbanktv The one gift #COVID19 has give me...
4  2020-07-25 12:27:08  25 July : Media Bulletin on Novel #CoronaVirus...

   ...     anger anticip     trust  surprise  positive  negative   sadness  \
0  ...  0.000000     0.0  0.000000  0.000000  0.000000  0.500000  0.000000
1  ...  0.000000     0.0  0.285714  0.000000  0.428571  0.000000  0.000000
2  ...  0.166667     0.0  0.000000  0.166667  0.166667  0.166667  0.166667
3  ...  0.000000     0.0  0.142857  0.142857  0.285714  0.000000  0.000000
4  ...  0.000000     0.0  0.000000  0.000000  0.000000  0.000000  0.000000

    disgust       joy  anticipation
0  0.500000  0.000000           NaN
1  0.000000  0.142857      0.142857
2  0.166667  0.000000           NaN
3  0.000000  0.285714      0.142857
4  0.000000  0.000000           NaN

[5 rows x 24 columns]
```

In [7]:

```
#first check how the Location is structured and if we can classify the data by

print(data_with_sentiment['user_location'])

# Fill NaN values with 0 before performing aggregation
data_with_sentiment.fillna(0, inplace=True)
```

```
0                        astroworld
1                      New York, NY
2                  Pewee Valley, KY
3               Stuck in the Middle
4                  Jammu and Kashmir
                      ...
179103              Ilorin, Nigeria
179104                      Ontario
179105                   🇨🇦 Canada
179106                New York City
179107    Aliwal North, South Africa
Name: user_location, Length: 179108, dtype: object
```

In [8]:

```
# Display the data with sentiment analysis results
print(data_with_sentiment.head())
```

```
         user_name         user_location  \
0           ᴛᴀᴍᴇᴛ            astroworld
1    Tom Basile 🇺🇸           New York, NY
2  Time4fisticuffs      Pewee Valley, KY
3      ethel mertz  Stuck in the Middle
4         DIPR-J&K     Jammu and Kashmir

                                  user_description         user_created  \
0  wednesday addams as a disney princess keepin i...  2017-05-26 05:46:42
1  Husband, Father, Columnist & Commentator. Auth...  2009-04-16 20:06:23
2  #Christian #Catholic #Conservative #Reagan #Re...  2009-02-28 18:57:41
3  #Browns #Indians #ClevelandProud #[]_[] #Cavs ...  2019-03-07 01:45:06
4  🖋Official Twitter handle of Department of Inf...  2017-02-12 06:45:15

   user_followers  user_friends  user_favourites  user_verified  \
0             624           950            18775          False
1            2253          1677               24           True
2            9275          9525             7254          False
3             197           987             1488          False
4          101009           168              101          False

                 date                                               text  \
0  2020-07-25 12:27:21  If I smelled the scent of hand sanitizers toda...
1  2020-07-25 12:27:17  Hey @Yankees @YankeesPR and @MLB - wouldn't it...
2  2020-07-25 12:27:14  @diane3443 @wdunlap @realDonaldTrump Trump nev...
3  2020-07-25 12:27:10  @brookbanktv The one gift #COVID19 has give me...
4  2020-07-25 12:27:08  25 July : Media Bulletin on Novel #CoronaVirus...

   ...     anger anticip     trust  surprise  positive  negative   sadness  \
0  ...  0.000000     0.0  0.000000  0.000000  0.000000  0.500000  0.000000
1  ...  0.000000     0.0  0.285714  0.000000  0.428571  0.000000  0.000000
2  ...  0.166667     0.0  0.000000  0.166667  0.166667  0.166667  0.166667
3  ...  0.000000     0.0  0.142857  0.142857  0.285714  0.000000  0.000000
4  ...  0.000000     0.0  0.000000  0.000000  0.000000  0.000000  0.000000

    disgust       joy  anticipation
0  0.500000  0.000000      0.000000
1  0.000000  0.142857      0.142857
2  0.166667  0.000000      0.000000
3  0.000000  0.285714      0.142857
4  0.000000  0.000000      0.000000

[5 rows x 24 columns]
```

In [9]:
```
# create a dataframe that aggregates and takes the mean of all the locations
sentiment_aggregated_df = data_with_sentiment.groupby('user_location', as_index
    'anger': 'mean',
    'anticip': 'mean',
    'trust': 'mean',
    'surprise': 'mean',
    'positive': 'mean',
    'negative': 'mean',
    'sadness': 'mean',
    'disgust': 'mean',
    'joy': 'mean',
    'anticipation': 'mean',
})
```

In [10]:
```
sentiment_aggregated_df['user_location']
```

Out[10]:
```
0                                0
1              \nsomewhere\n🖤
2
3              Cathlamet, Wa.
4                Canada 🇨🇦🍁🛶🌲
                 ...
26916                          🦄
26917            🦄🌈❤️🧡💛💚💙💜🌈🦄
26918                 🦅💛💛💙❤️🏔️
26919               🦠Grounded
26920                          🪐
Name: user_location, Length: 26921, dtype: object
```

In [11]:
```
# Check if the argument contains only letters using regex
def contains_only_letters(s):
    return bool(re.match(r'^[a-zA-Z\s]+$', str(s)))

# Filter rows and take out locations with numbers and emojis - clean
filtered_senti_agg_df = sentiment_aggregated_df[sentiment_aggregated_df['user_l
print(filtered_senti_agg_df)
```

```
                     user_location    anger  anticip     trust  surprise  \
2                                   0.07684      0.0  0.050577  0.030339
5                        WorldWide  0.00000      0.0  0.000000  0.000000
8                         Fujairah  0.00000      0.0  0.166667  0.166667
9        NO investment advice given  0.00000      0.0  0.000000  0.000000
10             Shropshire England   0.00000      0.0  0.125000  0.083333
...                             ...      ...      ...       ...       ...
26114                     your mom  0.00000      0.0  0.000000  0.000000
26116              your mums house  0.00000      0.0  0.000000  0.000000
26117                     yourmama  0.25000      0.0  0.000000  0.000000
26121                          yyc  0.00000      0.0  0.250000  0.000000
26123                       zagreb  0.00000      0.0  0.250000  0.125000

       positive  negative   sadness   disgust       joy  anticipation
2      0.211436  0.145599  0.076587  0.041450  0.047042      0.093506
5      0.000000  0.000000  0.000000  0.000000  0.000000      0.000000
8      0.166667  0.000000  0.000000  0.166667  0.166667      0.166667
9      0.755556  0.111111  0.000000  0.000000  0.000000      0.133333
10     0.208333  0.333333  0.000000  0.000000  0.083333      0.083333
...         ...       ...       ...       ...       ...           ...
26114  0.000000  0.500000  0.000000  0.000000  0.000000      0.000000
26116  0.000000  1.000000  0.000000  0.000000  0.000000      0.000000
26117  0.000000  0.250000  0.166667  0.166667  0.000000      0.000000
26121  0.250000  0.500000  0.000000  0.000000  0.000000      0.000000
26123  0.250000  0.000000  0.000000  0.000000  0.125000      0.250000

[8286 rows x 11 columns]
```

In [12]:

```
# Filter rows based on the condition
us_filtered_df = filtered_senti_agg_df[filtered_senti_agg_df['user_location'].s
print(us_filtered_df)
```

```
              user_location     anger  anticip     trust  surprise  positive
\
3262   Big Island of Hawaii  0.200000      0.0  0.000000  0.000000  0.000000
9128                 Hawaii  0.015686      0.0  0.133578  0.073529  0.208088
9135       Hawaiian Islands  0.000000      0.0  0.000000  0.000000  0.166667
20422       State of Hawaii  0.000000      0.0  0.333333  0.000000  0.333333

       negative   sadness   disgust       joy  anticipation
3262   0.200000  0.200000  0.200000  0.000000      0.000000
9128   0.128922  0.027451  0.015686  0.073284      0.063480
9135   0.111111  0.111111  0.000000  0.000000      0.166667
20422  0.166667  0.000000  0.000000  0.000000      0.166667
```

In [13]:

```
states = [
    'Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California', 'Colorado', 'Conr
    'Delaware', 'Florida', 'Georgia', 'Hawaii', 'Idaho', 'Illinois', 'Indiana',
    'Kansas', 'Kentucky', 'Louisiana', 'Maine', 'Maryland', 'Massachusetts', 'N
    'Minnesota', 'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada', 'N
    'New Jersey', 'New Mexico', 'New York', 'North Carolina', 'North Dakota', '
    'Oregon', 'Pennsylvania', 'Rhode Island', 'South Carolina', 'South Dakota',
    'Texas', 'Utah', 'Vermont', 'Virginia', 'Washington', 'West Virginia', 'Wis
]

# Create a a filter mask to get rows of that ocntain every state only
state_mask = filtered_senti_agg_df.apply(lambda row: any(state in row['user_loc

us_filtered_df = filtered_senti_agg_df[state_mask]
print(us_filtered_df)
```

```
                     user_location     anger  anticip     trust  surprise  \
11       Sunny Southern California  0.053030      0.0  0.075758  0.000000
24         A Life Long  New Yorker  0.250000      0.0  0.000000  0.000000
27                         Alabama  0.000000      0.0  0.500000  0.000000
54          Colorado Plains Native  0.038889      0.0  0.059722  0.045833
66                         Florida  0.125000      0.0  0.125000  0.000000
...                            ...       ...      ...       ...       ...
25412      make Florida BLUE again  0.200000      0.0  0.000000  0.200000
25545           north central Ohio  0.000000      0.0  0.000000  0.000000
25634     patch of blue in Florida  0.142857      0.0  0.000000  0.000000
25770              small town Texas  0.000000      0.0  0.000000  0.000000
25830          southern California  0.000000      0.0  0.000000  0.000000

       positive  negative   sadness   disgust       joy  anticipation
11     0.242424  0.113636  0.015152  0.022727  0.015152      0.045455
24     0.000000  0.250000  0.000000  0.250000  0.000000      0.000000
27     0.500000  0.000000  0.000000  0.000000  0.000000      0.000000
54     0.122222  0.223611  0.070139  0.013889  0.045833      0.020833
66     0.125000  0.250000  0.125000  0.125000  0.000000      0.000000
...         ...       ...       ...       ...       ...           ...
25412  0.000000  0.200000  0.200000  0.200000  0.000000      0.000000
25545  0.000000  0.000000  0.000000  0.000000  0.000000      0.000000
25634  0.000000  0.142857  0.071429  0.071429  0.000000      0.000000
25770  0.000000  0.000000  0.000000  0.000000  0.000000      0.000000
25830  0.000000  0.500000  0.000000  0.000000  0.000000      0.000000

[542 rows x 11 columns]
```

In [14]:

```
#here we will take the average and create a new DF with 50 rows and average ser
# Iterate through each state and aggregate scores
import warnings
warnings.filterwarnings('ignore')
agg_us_scores_df = pd.DataFrame(columns=['user_location'] + us_filtered_df.colu
for state in states:
    state_rows = us_filtered_df[us_filtered_df['user_location'].str.contains(st

    if not state_rows.empty:
        state_avg_scores = state_rows.mean()
        state_avg_scores['user_location'] = state
        agg_us_scores_df = agg_us_scores_df.append(state_avg_scores, ignore_inc
```

In [15]:

```
'''
create an overall column that classifies the state as negative or positive:

sum the columns of anger, anticip, negative, sadness, disgust, anticipation for
sum the columns od joy, surprisem, trust, and positive for positive
check which sum is greater, if negative is greater write negative for column ot
do this for all 50 rows
'''
```

Out[15]:

```
'\ncreate an overall column that classifies the state as negative or positiv
e:\n\nsum the columns of anger, anticip, negative, sadness, disgust, anticipat
ion for negative \nsum the columns od joy, surprisem, trust, and positive for
positive \ncheck which sum is greater, if negative is greater write negative f
or column otherwise write positive,\ndo this for all 50 rows \n'
```

In [16]:

```
agg_us_scores_df['negative_overall'] = agg_us_scores_df[['anger', 'anticip', 'n
agg_us_scores_df['positive_overall'] = agg_us_scores_df[['trust', 'surprise', '
#Classify rows based on overall sentiment comparison
agg_us_scores_df['overall'] = agg_us_scores_df.apply(lambda row: 'positive' if
```

In [17]:
```
'''
Below create a plot of the US map
plot the strongest sentiment for the state
Create bar graphs by state and create a bar graph that compares positive to neg
anger   anticip trust   surprise        positive        negative        sadness
'''
```

Out[17]:
```
'\nBelow create a plot of the US map \nplot the strongest sentiment for the st
ate \nCreate bar graphs by state and create a bar graph that compares positive
to negative \nanger\tanticip\ttrust\tsurprise\tpositive\tnegative\tsadness\tdi
sgust\tjoy\tanticipation\n'
```

In [18]:
```
import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt
```

In [19]:
```
#find the column with the strongest sentiment

# Calculate strongest sentiment for each state
sentiment_columns = ['anger', 'anticip', 'trust', 'surprise', 'positive', 'nega
agg_us_scores_df['strongest_sentiment'] = agg_us_scores_df[sentiment_columns].i

agg_us_scores_df
```

Out[19]:

	user_location	anger	anticip	trust	surprise	positive	negative	sadness	dis
0	Alabama	0.002670	0.0	0.200302	0.018384	0.188794	0.134620	0.010879	0.03
1	Alaska	0.056447	0.0	0.119493	0.004456	0.054483	0.065191	0.021122	0.01
2	Arizona	0.114036	0.0	0.085088	0.001273	0.159430	0.133965	0.011533	0.00
3	Arkansas	0.021970	0.0	0.066064	0.004631	0.151812	0.028840	0.009107	0.00
4	California	0.027277	0.0	0.115297	0.017014	0.201627	0.122074	0.037427	0.02
5	Colorado	0.028601	0.0	0.237927	0.004150	0.204377	0.100326	0.051442	0.01
6	Connecticut	0.007165	0.0	0.099459	0.001134	0.210038	0.150347	0.063382	0.00
7	Delaware	0.073769	0.0	0.052800	0.001420	0.081636	0.176001	0.007819	0.04
8	Florida	0.033622	0.0	0.061704	0.013890	0.175530	0.135855	0.048314	0.03
9	Georgia	0.010267	0.0	0.124279	0.008961	0.171858	0.188340	0.066592	0.00
10	Hawaii	0.053922	0.0	0.116728	0.018382	0.177022	0.151675	0.084641	0.05
11	Idaho	0.031507	0.0	0.093545	0.003787	0.116440	0.262762	0.104753	0.00
12	Illinois	0.032679	0.0	0.140503	0.005048	0.150130	0.082832	0.076252	0.01
13	Indiana	0.042436	0.0	0.129132	0.045969	0.189190	0.088760	0.042905	0.03
14	Iowa	0.011574	0.0	0.334606	0.007581	0.197743	0.188397	0.009230	0.00
15	Kansas	0.014974	0.0	0.111481	0.024518	0.166489	0.049694	0.022098	0.00
16	Kentucky	0.005673	0.0	0.074542	0.002339	0.141167	0.129339	0.171923	0.00
17	Louisiana	0.012309	0.0	0.074782	0.069608	0.103050	0.050545	0.014052	0.00
18	Maine	0.010439	0.0	0.282134	0.046987	0.128494	0.051117	0.011296	0.00
19	Maryland	0.093084	0.0	0.039266	0.019048	0.050959	0.253302	0.146108	0.00
20	Massachusetts	0.060565	0.0	0.106015	0.042534	0.186096	0.148118	0.049447	0.00
21	Michigan	0.034634	0.0	0.119701	0.049800	0.121993	0.164133	0.002338	0.05
22	Minnesota	0.001832	0.0	0.134603	0.000641	0.128624	0.098134	0.039345	0.03
23	Mississippi	0.050397	0.0	0.068545	0.001852	0.247407	0.081508	0.019471	0.01
24	Missouri	0.016071	0.0	0.082301	0.019481	0.342878	0.049788	0.026578	0.02
25	Montana	0.000000	0.0	0.000000	0.000000	0.250000	0.000000	0.000000	0.00
26	Nebraska	0.048077	0.0	0.036538	0.000000	0.361538	0.019231	0.009615	0.00
27	Nevada	0.002915	0.0	0.034742	0.000000	0.089164	0.202381	0.030612	0.00
28	New Hampshire	0.003961	0.0	0.033918	0.023569	0.568994	0.011151	0.011151	0.00
29	New Jersey	0.016401	0.0	0.137736	0.000814	0.186689	0.130169	0.031961	0.00
30	New Mexico	0.023208	0.0	0.019362	0.007998	0.125772	0.049781	0.017614	0.01
31	New York	0.049371	0.0	0.118170	0.029024	0.185814	0.133903	0.056896	0.03
32	North Carolina	0.065729	0.0	0.010745	0.057531	0.049436	0.138481	0.027730	0.00
33	North Dakota	0.077976	0.0	0.097222	0.023810	0.097222	0.258532	0.077976	0.05

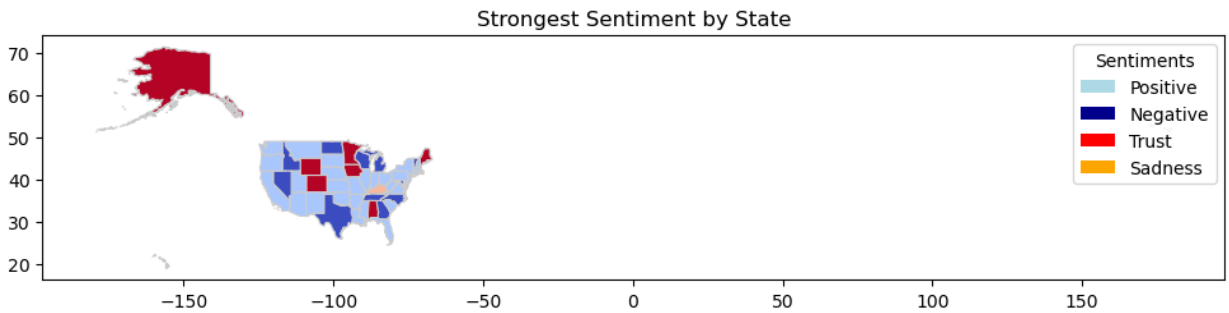
	user_location	anger	anticip	trust	surprise	positive	negative	sadness	dis
34	Ohio	0.015935	0.0	0.127872	0.024574	0.288905	0.117548	0.031486	0.01
35	Oklahoma	0.028240	0.0	0.172890	0.069174	0.218338	0.051587	0.021531	0.01
36	Oregon	0.020957	0.0	0.036247	0.070957	0.201390	0.118935	0.043163	0.02
37	Pennsylvania	0.003516	0.0	0.097885	0.011888	0.132317	0.021779	0.004439	0.00
38	Rhode Island	0.072727	0.0	0.091017	0.019048	0.188095	0.228788	0.047727	0.04
39	South Carolina	0.009238	0.0	0.109800	0.007133	0.275704	0.058328	0.013624	0.00
40	South Dakota	0.000000	0.0	0.166667	0.000000	0.333333	0.166667	0.166667	0.00
41	Tennessee	0.086728	0.0	0.065983	0.000000	0.073038	0.175386	0.046197	0.01
42	Texas	0.030502	0.0	0.072974	0.051478	0.147305	0.188573	0.060635	0.02
43	Utah	0.001471	0.0	0.034475	0.112500	0.212857	0.072381	0.068347	0.00
44	Vermont	0.012488	0.0	0.066364	0.048368	0.081932	0.286880	0.022394	0.00
45	Virginia	0.054642	0.0	0.088879	0.029125	0.158464	0.076529	0.043198	0.03
46	Washington	0.044714	0.0	0.147548	0.034992	0.264687	0.167897	0.043517	0.01
47	West Virginia	0.061111	0.0	0.137500	0.000000	0.279167	0.088194	0.098611	0.04
48	Wisconsin	0.045635	0.0	0.027778	0.020833	0.045139	0.330159	0.051472	0.04
49	Wyoming	0.000000	0.0	0.642857	0.071429	0.071429	0.071429	0.071429	0.00

In [21]:
```
# Load GeoJSON file containing state geometries
geojson_file = "/Users/gurnoorvirdi/Desktop/NLP- social sciences/NLP Final Proj
us_map = gpd.read_file(geojson_file)

# us_map the GeoDataFrame's structure
print(us_map.head())
```

```
        GEO_ID STATE         NAME LSAD  CENSUSAREA  \
0  0400000US04    04      Arizona       113594.084
1  0400000US05    05     Arkansas        52035.477
2  0400000US06    06   California       155779.220
3  0400000US08    08     Colorado       103641.888
4  0400000US09    09  Connecticut         4842.355

                                            geometry
0  POLYGON ((-112.53859 37.00067, -112.53454 37.0...
1  POLYGON ((-94.04296 33.01922, -94.04304 33.079...
2  MULTIPOLYGON (((-120.24848 33.99933, -120.2473...
3  POLYGON ((-107.31779 41.00296, -107.00061 41.0...
4  POLYGON ((-72.39743 42.03330, -72.19883 42.030...
```

In [22]:
```
from matplotlib.patches import Patch
# Merge sentiment data with US map data
merged_data = us_map.merge(agg_us_scores_df, how='left', left_on='NAME', right_
merged_data
# Plot the US map with labeled states
fig, ax = plt.subplots(figsize=(12, 10))
merged_data.plot(column='strongest_sentiment', cmap='coolwarm', linewidth=0.8,

color_map = {
```

```
    'Positive': 'lightblue',
    'Negative': 'darkblue',
    'Trust': 'red',
    'Sadness': 'orange'
}

# Add legend
legend_elements = [Patch(facecolor=color_map[sentiment], label=sentiment) for s
ax.legend(handles=legend_elements, title='Sentiments')


ax.set_title('Strongest Sentiment by State')
plt.show()
```



In [23]:
```
merged_data
```

Out[23]:

	GEO_ID	STATE	NAME	LSAD	CENSUSAREA	geometry	user_location
0	0400000US04	04	Arizona		113594.084	POLYGON ((-112.5385937.00067, -112.5345437.0...	Arizona
1	0400000US05	05	Arkansas		52035.477	POLYGON ((-94.0429633.01922, -94.0430433.079...	Arkansas
2	0400000US06	06	California		155779.220	MULTIPOLYGON (((-120.2484833.99933, -120.2473...	California
3	0400000US08	08	Colorado		103641.888	POLYGON ((-107.3177941.00296, -107.0006141.0...	Colorado
4	0400000US09	09	Connecticut		4842.355	POLYGON ((-72.3974342.03330, -72.1988342.030...	Connecticut
5	0400000US11	11	District of Columbia		61.048	POLYGON ((-77.0329938.83950, -77.0317038.850...	NaN
6	0400000US13	13	Georgia		57513.485	POLYGON ((-84.8104834.98761, -84.8091834.987...	Georgia
7	0400000US15	15	Hawaii		6422.628	MULTIPOLYGON (((-155.7782320.24574, -155.7727...	Hawaii
8	0400000US17	17	Illinois		55518.930	POLYGON ((-89.3660342.50027, -89.3615642.500...	Illinois
9	0400000US18	18	Indiana		35826.109	POLYGON ((-84.8041240.35276, -84.8039240.310...	Indiana
10	0400000US22	22	Louisiana		43203.905	MULTIPOLYGON (((-88.8650729.75271, -88.88975 ...	Louisiana
11	0400000US27	27	Minnesota		79626.743	POLYGON ((-92.19150	Minnesota

	GEO_ID	STATE	NAME	LSAD	CENSUSAREA	geometry	user_location
						46.67259, -92.19715 46.663...	
12	0400000US28	28	Mississippi		46923.274	MULTIPOLYGON (((-89.09562 30.23177, -89.07726 ...	Mississippi
13	0400000US30	30	Montana		145545.801	POLYGON ((-111.04427 45.00135, -111.05621 44.9...	Montana
14	0400000US35	35	New Mexico		121298.148	POLYGON ((-105.99800 32.00233, -106.09976 32.0...	New Mexico
15	0400000US38	38	North Dakota		69000.798	POLYGON ((-100.51195 45.94365, -100.62768 45.9...	North Dakota
16	0400000US40	40	Oklahoma		68594.921	POLYGON ((-100.00038 34.74636, -100.00038 34.7...	Oklahoma
17	0400000US42	42	Pennsylvania		44742.703	POLYGON ((-79.47666 39.72108, -79.60822 39.721...	Pennsylvania
18	0400000US47	47	Tennessee		41234.896	POLYGON ((-83.47211 36.59728, -83.27630 36.598...	Tennessee
19	0400000US51	51	Virginia		39490.086	MULTIPOLYGON (((-75.24227 38.02721, -75.29687 ...	Virginia
20	0400000US72	72	Puerto Rico		3423.775	MULTIPOLYGON (((-65.28076 18.28827, -65.28327 ...	NaN
21	0400000US10	10	Delaware		1948.543	MULTIPOLYGON (((-75.56493 39.58325, -75.57627 ...	Delaware
22	0400000US54	54	West Virginia		24038.210	POLYGON ((-78.57190 39.03200, -78.56584 39.026...	West Virginia

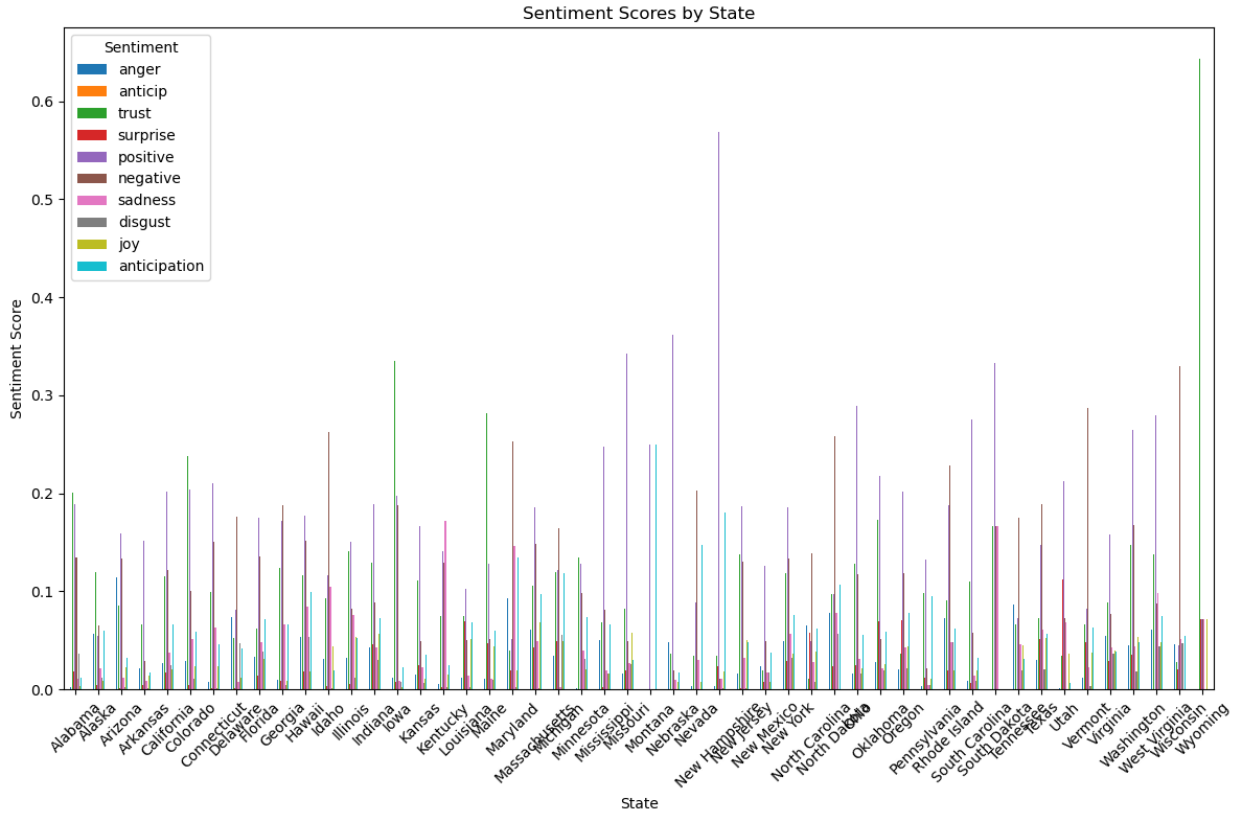
	GEO_ID	STATE	NAME	LSAD	CENSUSAREA	geometry	user_location
23	0400000US55	55	Wisconsin		54157.805	MULTIPOLYGON (((-90.45668 47.01674, -90.45530 ...	Wisconsin
24	0400000US56	56	Wyoming		97093.141	POLYGON ((-104.05508 43.93653, -104.05510 43.8...	Wyoming
25	0400000US01	01	Alabama		50645.326	MULTIPOLYGON (((-88.12466 30.28364, -88.08681 ...	Alabama
26	0400000US02	02	Alaska		570640.950	MULTIPOLYGON (((-162.25503 54.97835, -162.2496...	Alaska
27	0400000US12	12	Florida		53624.759	MULTIPOLYGON (((-80.25058 25.34193, -80.25492 ...	Florida
28	0400000US16	16	Idaho		82643.117	POLYGON ((-111.04897 44.47407, -111.04919 44.4...	Idaho
29	0400000US20	20	Kansas		81758.717	POLYGON ((-99.54112 36.99957, -99.55807 36.999...	Kansas
30	0400000US24	24	Maryland		9707.241	MULTIPOLYGON (((-76.04837 38.12055, -76.05681 ...	Maryland
31	0400000US34	34	New Jersey		7354.220	POLYGON ((-74.90024 40.07715, -74.83801 40.100...	New Jersey
32	0400000US37	37	North Carolina		48617.905	MULTIPOLYGON (((-75.75377 35.19961, -75.74522 ...	North Carolina
33	0400000US45	45	South Carolina		30060.696	POLYGON ((-82.21622 35.19604, -82.19548 35.194...	South Carolina
34	0400000US53	53	Washington		66455.521	MULTIPOLYGON (((-122.39735 47.91240, -122.4192...	Washington

	GEO_ID	STATE	NAME	LSAD	CENSUSAREA	geometry	user_location
35	0400000US50	50	Vermont		9216.657	POLYGON ((-72.45852 42.72685, -72.86418 42.737...	Vermont
36	0400000US49	49	Utah		82169.620	POLYGON ((-111.04669 42.00157, -111.04640 41.5...	Utah
37	0400000US19	19	Iowa		55857.130	POLYGON ((-91.16306 42.98678, -91.14556 42.907...	Iowa
38	0400000US21	21	Kentucky		39486.338	MULTIPOLYGON (((-89.53910 36.49820, -89.56034 ...	Kentucky
39	0400000US23	23	Maine		30842.923	MULTIPOLYGON (((-69.30791 43.77377, -69.30675 ...	Maine
40	0400000US25	25	Massachusetts		7800.058	MULTIPOLYGON (((-70.82100 41.58727, -70.82174 ...	Massachusetts
41	0400000US26	26	Michigan		56538.901	MULTIPOLYGON (((-85.56644 45.76022, -85.54956 ...	Michigan
42	0400000US29	29	Missouri		68741.522	POLYGON ((-89.54501 36.33681, -89.56044 36.337...	Missouri
43	0400000US31	31	Nebraska		76824.171	POLYGON ((-104.05283 41.69795, -104.05277 41.7...	Nebraska
44	0400000US32	32	Nevada		109781.180	POLYGON ((-114.04655 40.11693, -114.04713 39.9...	Nevada
45	0400000US33	33	New Hampshire		8952.651	POLYGON ((-70.81955 43.12323, -70.78411 43.098...	New Hampshire
46	0400000US36	36	New York		47126.399	MULTIPOLYGON (((-73.77336	New York

	GEO_ID	STATE	NAME	LSAD	CENSUSAREA	geometry	user_location
						40.85945, -73.77055 ...	
47	0400000US39	39	Ohio		40860.694	MULTIPOLYGON (((-82.70021 41.61219, -82.69112 ...	Ohio
48	0400000US41	41	Oregon		95988.013	POLYGON ((-121.90827 45.65440, -121.90086 45.6...	Oregon
49	0400000US44	44	Rhode Island		1033.814	MULTIPOLYGON (((-71.38359 41.46478, -71.38928 ...	Rhode Island
50	0400000US46	46	South Dakota		75811.000	POLYGON ((-104.05510 43.85348, -104.05508 43.9...	South Dakota
51	0400000US48	48	Texas		261231.711	MULTIPOLYGON (((-96.83003 28.11184, -96.82705 ...	Texas

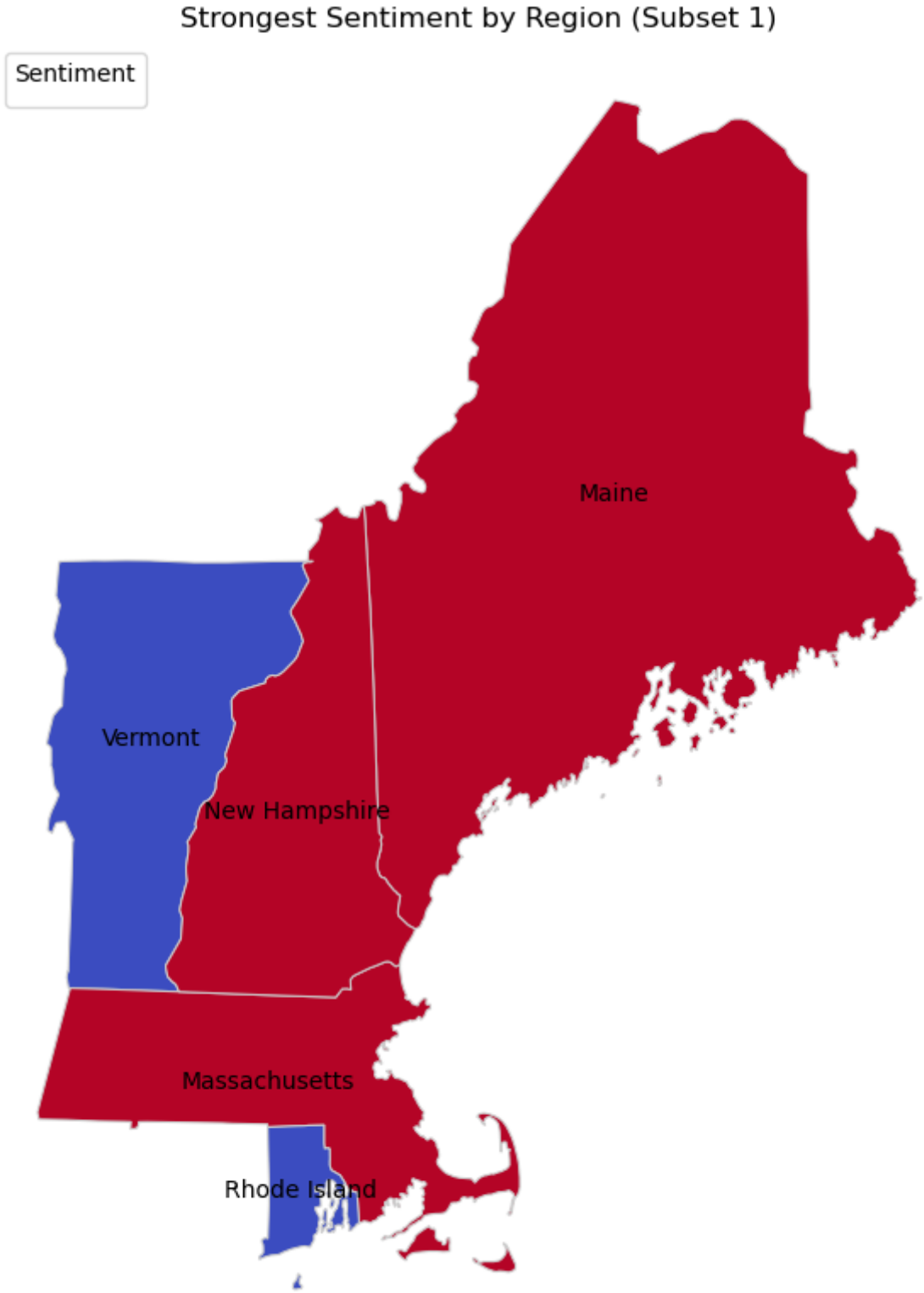
52 rows × 21 columns

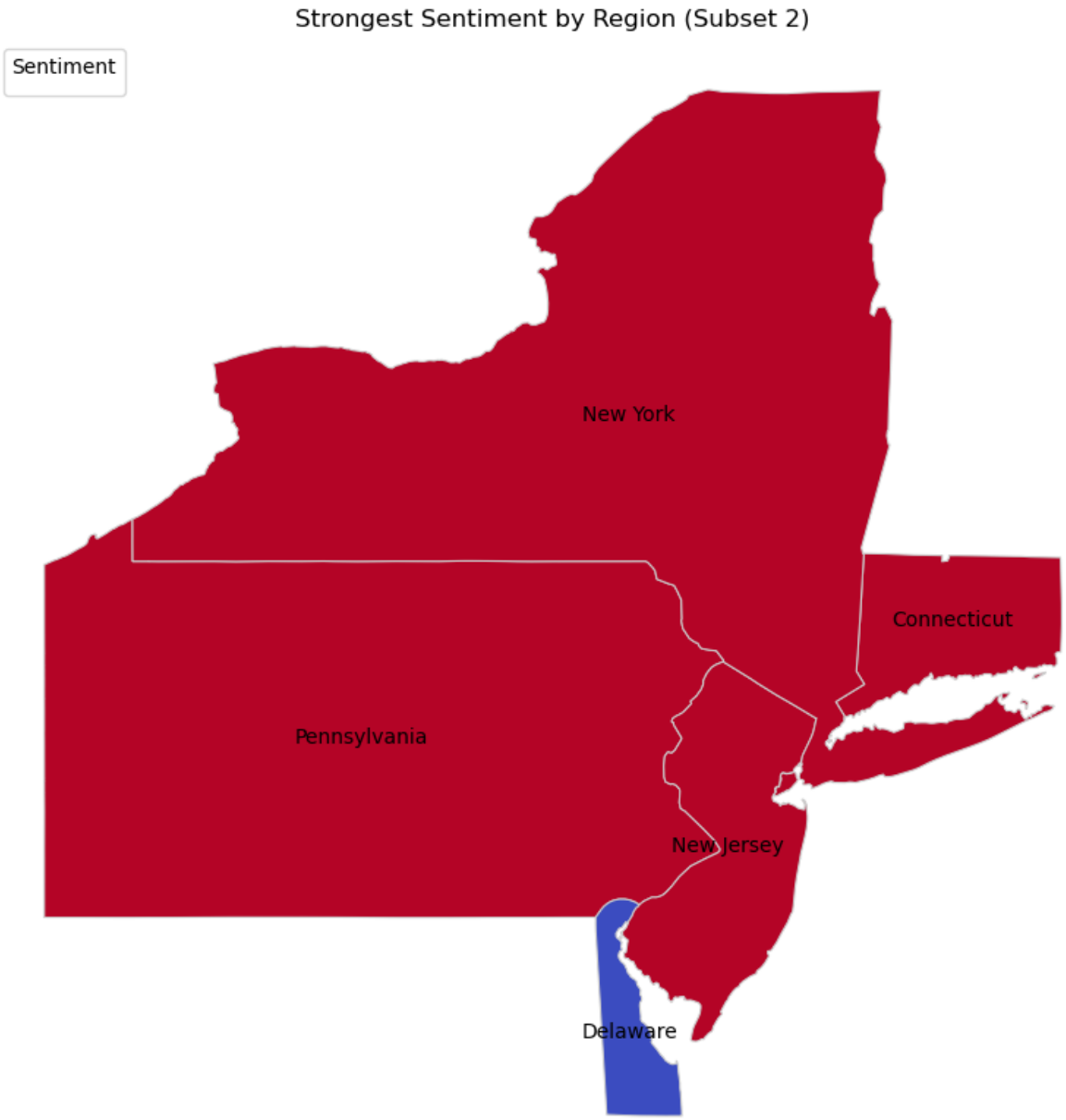
In [24]:
```
#plot bar graphs by state


# Plot bar graph by state for sentiment scores
agg_us_scores_df.set_index('user_location')[sentiment_columns].plot(kind='bar',
plt.title('Sentiment Scores by State')
plt.xlabel('State')
plt.ylabel('Sentiment Score')
plt.xticks(rotation=45)
plt.legend(title='Sentiment')
plt.tight_layout()

plt.show()
```

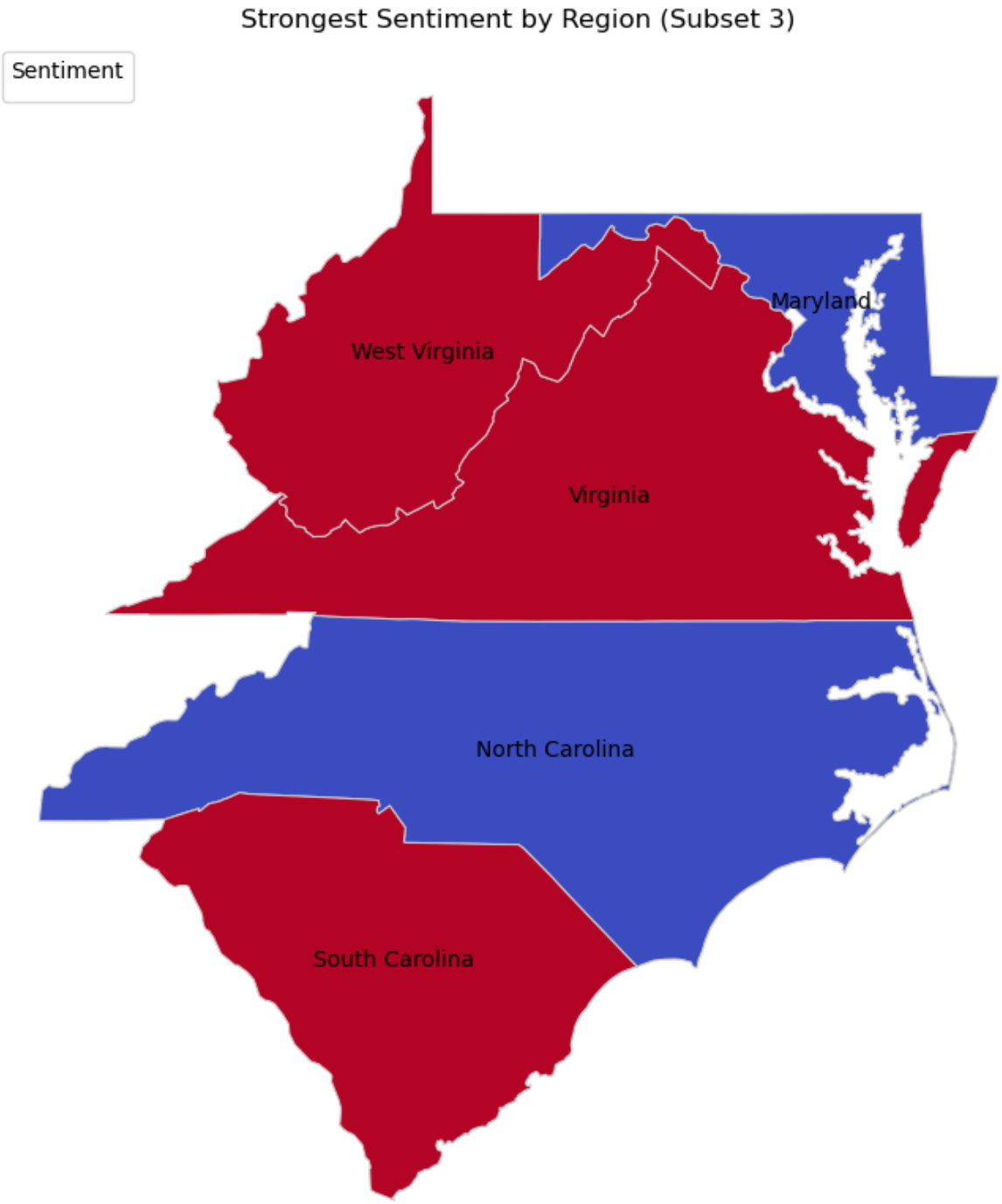


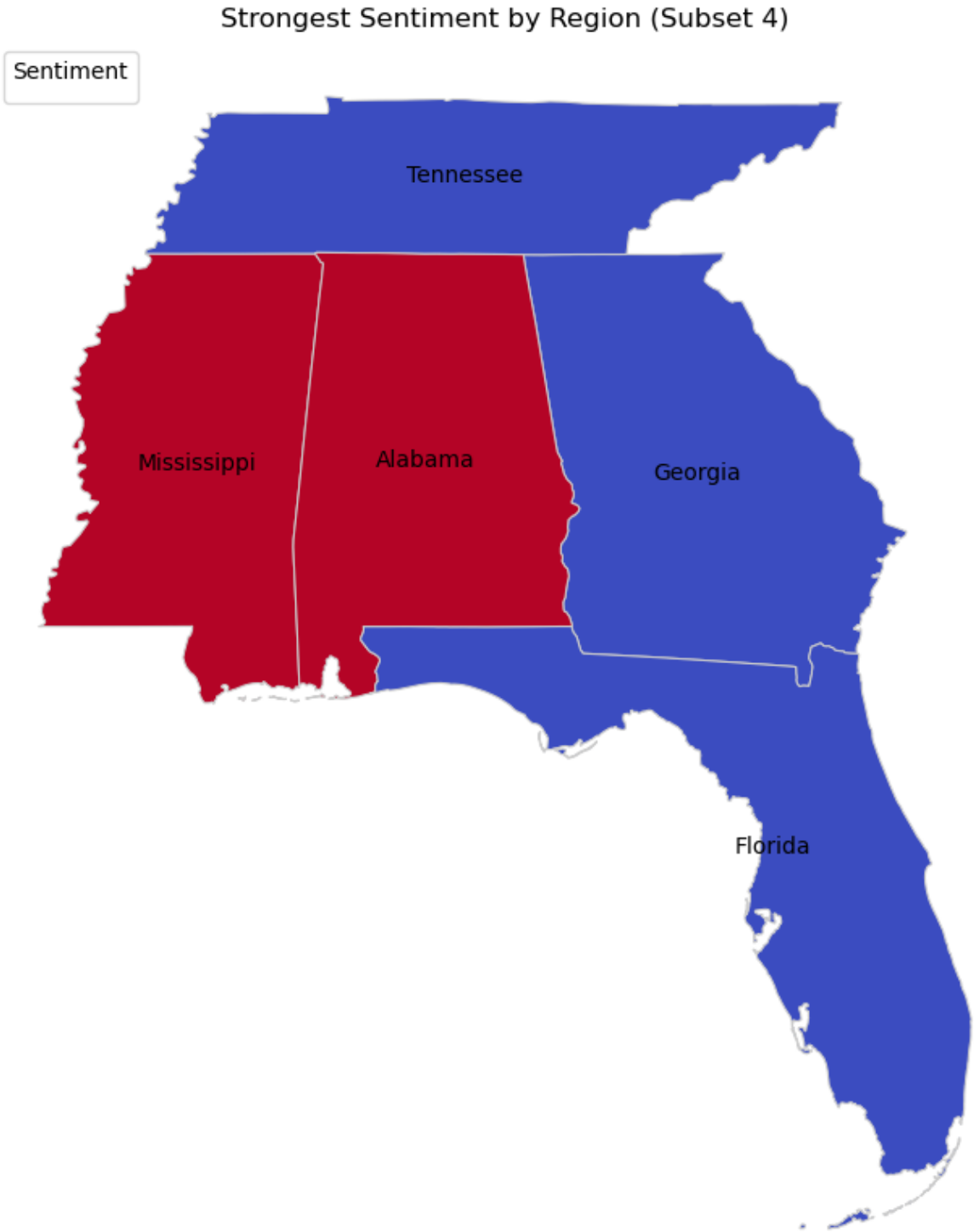
In [31]:
```
# Define subsets of states starting from the east coast
subset_states_list = [
    ['Maine', 'New Hampshire', 'Vermont', 'Massachusetts', 'Rhode Island'],
    ['Connecticut', 'New York', 'New Jersey', 'Pennsylvania', 'Delaware'],
    ['Maryland', 'Virginia', 'West Virginia', 'North Carolina', 'South Carolina
    ['Georgia', 'Florida', 'Alabama', 'Mississippi', 'Tennessee'],
    ['Kentucky', 'Ohio', 'Michigan', 'Indiana', 'Illinois'],
    ['Wisconsin', 'Minnesota', 'Iowa', 'Missouri', 'Arkansas'],
    ['Louisiana', 'Texas', 'Oklahoma', 'Kansas', 'Nebraska'],
    ['South Dakota', 'North Dakota', 'Montana', 'Wyoming', 'Colorado'],
    ['New Mexico', 'Arizona', 'Utah', 'Idaho', 'Nevada'],
    ['Washington', 'Oregon', 'California', 'Alaska', 'Hawaii']
]

# Legend labels
legend_labels = {
    'Anger': 'Anger',
    'Negative': 'Negative',
    'Positive': 'Positive',
    'Joy': 'Joy',
    'Trust': 'Trust',
    'Anticipation': 'Anticipation'
}

# Loop through each subset and create choropleth maps
for i, subset_states in enumerate(subset_states_list):
    sub_gdf = us_map[us_map['NAME'].isin(subset_states)]
    merged_data_subset = pd.merge(sub_gdf, agg_us_scores_df[agg_us_scores_df['u

    fig, ax = plt.subplots(1, 1, figsize=(15, 10))
    merged_data_subset.plot(column='overall', cmap='coolwarm', linewidth=0.8, a

    plt.title(f"Strongest Sentiment by Region (Subset {i+1})")
```

```
    ax.legend(title='Sentiment', loc='upper left', labels=[legend_labels])

    for index, row in sub_gdf.iterrows():
        x, y = row['geometry'].centroid.x, row['geometry'].centroid.y
        state_name = row['NAME']  # Assuming your GeoJSON attribute is 'state_n
        ax.text(x, y, state_name, fontsize=10, ha='center', va='center')

    ax.set_axis_off()

    # Show the plot or save it to a file
    plt.show()
```

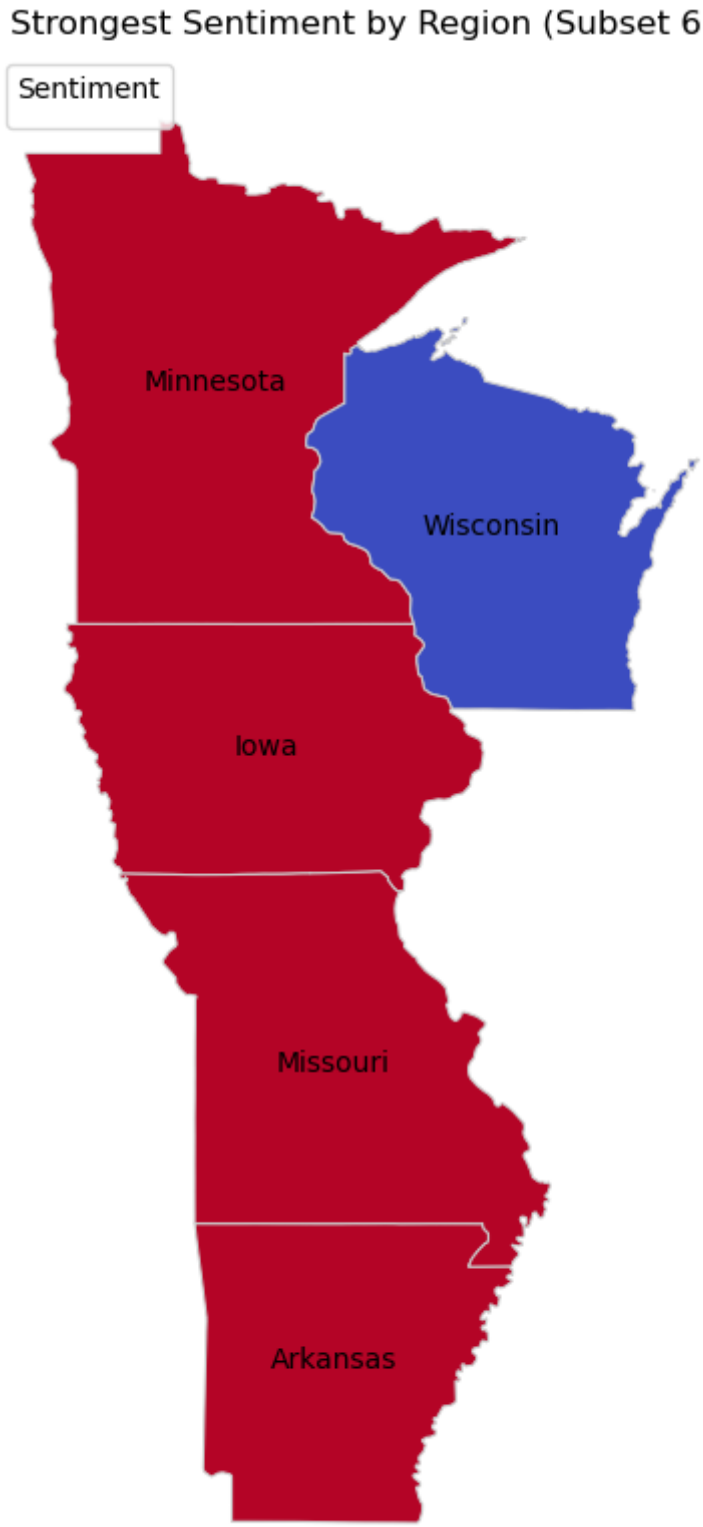

Strongest Sentiment by Region (Subset 1)
Sentiment
Maine
Vermont
New Hampshire
Massachusetts
Rhode Island

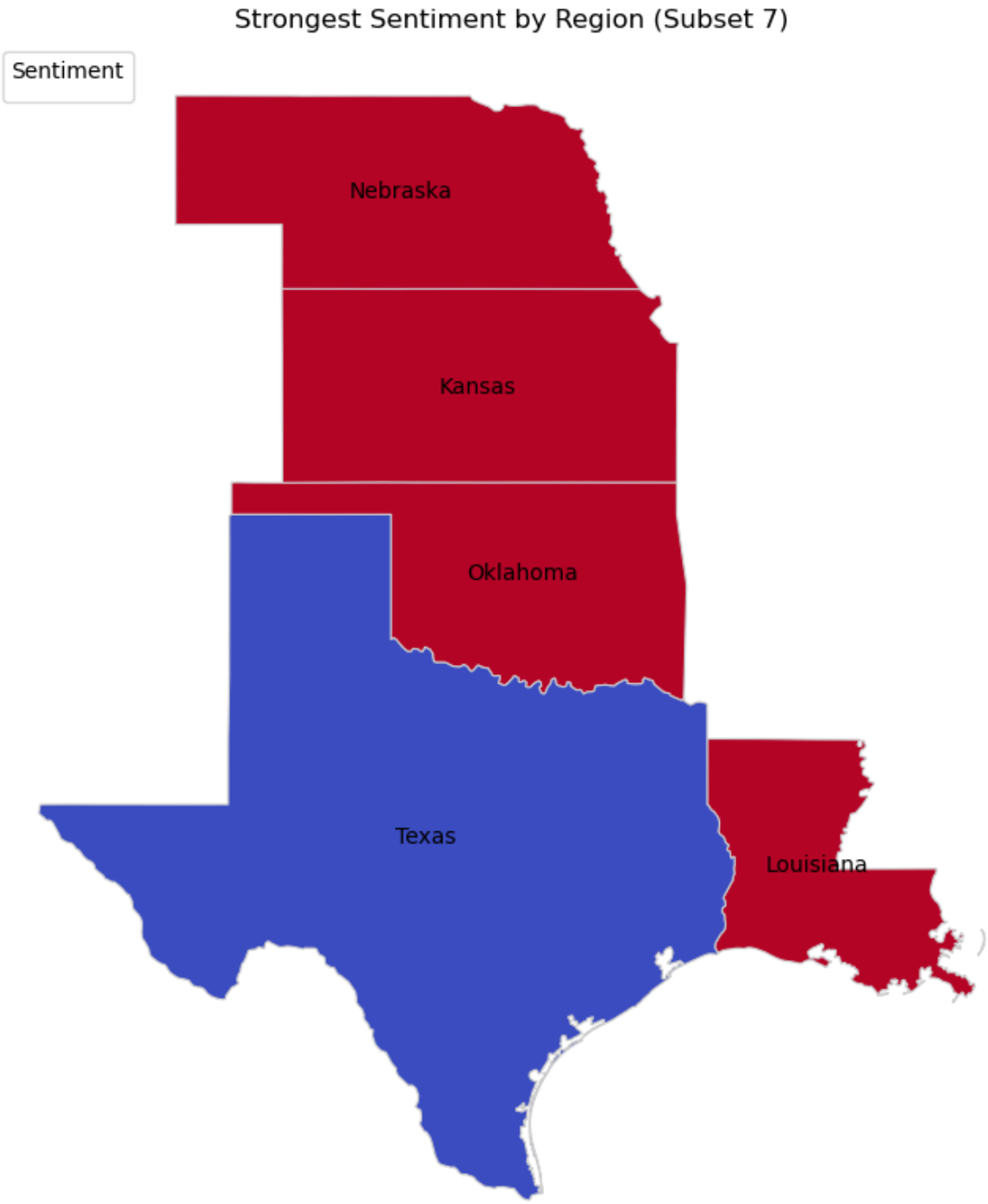

Strongest Sentiment by Region (Subset 2)
Sentiment
New York
Connecticut
Pennsylvania
New Jersey
Delaware

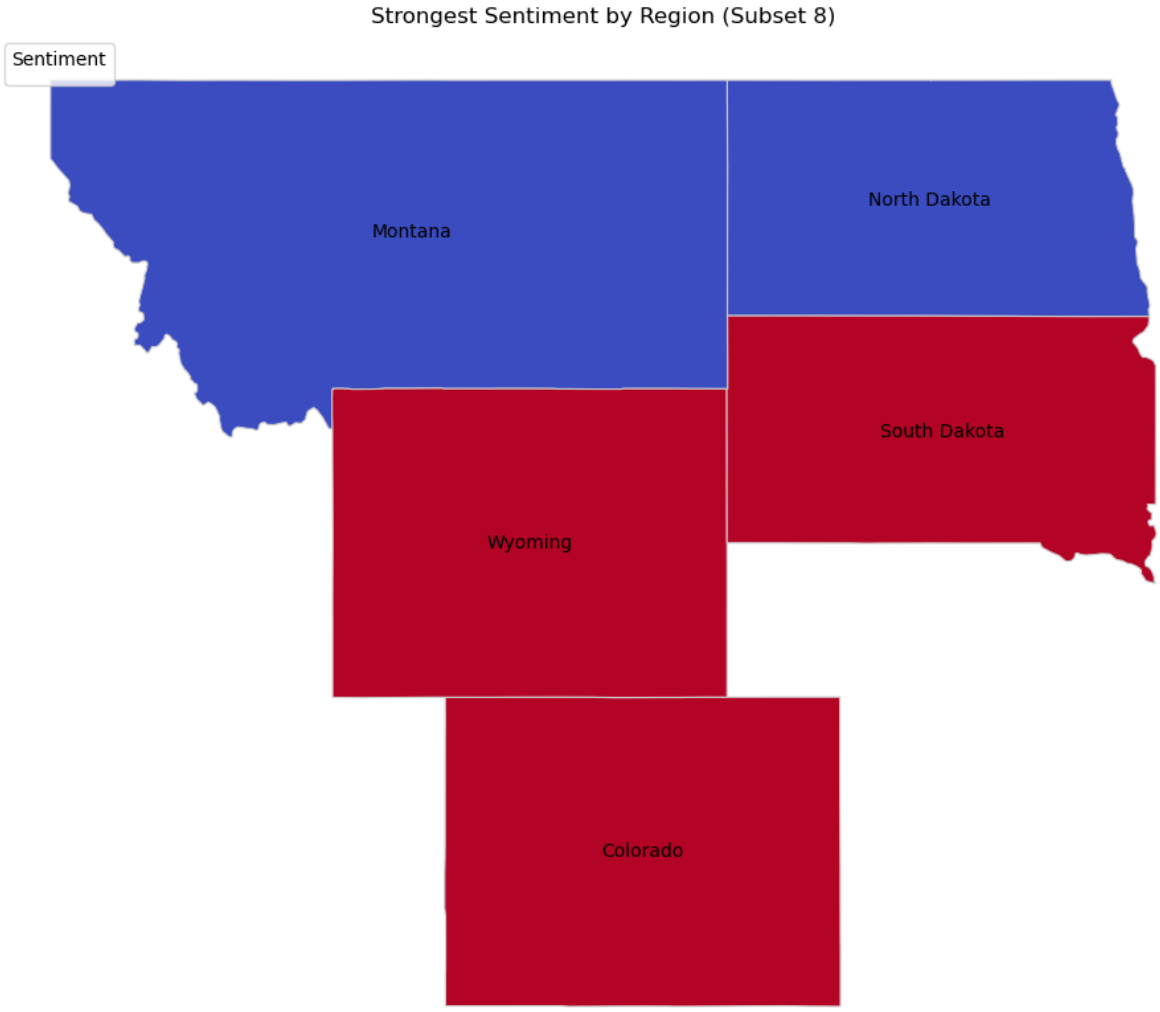

Strongest Sentiment by Region (Subset 3)
Sentiment
Maryland
West Virginia
Virginia
North Carolina
South Carolina

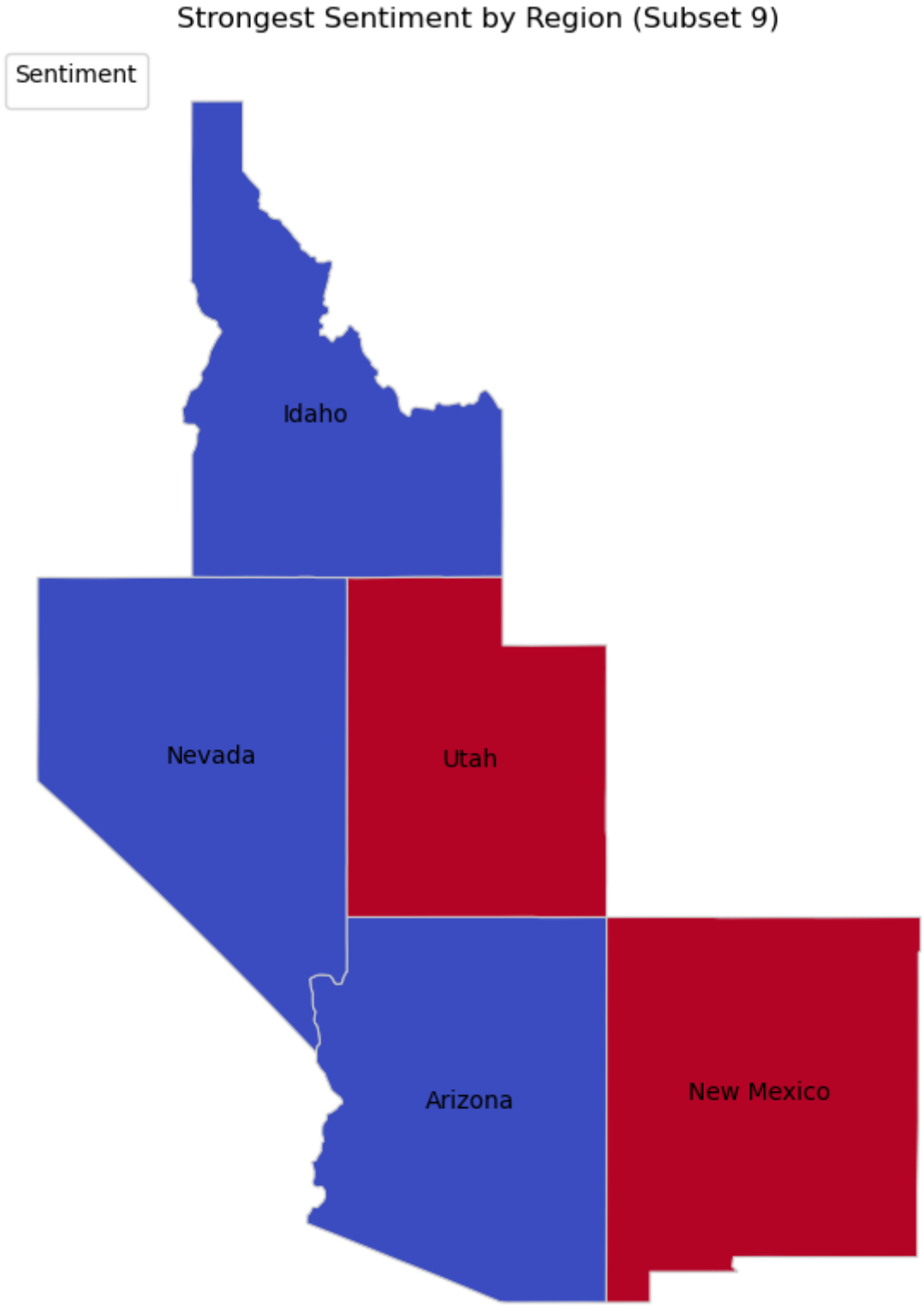

Strongest Sentiment by Region (Subset 4)
Sentiment
Tennessee
Mississippi
Alabama
Georgia
Florida


Strongest Sentiment by Region (Subset 5)
Sentiment
Michigan
Illinois
Indiana
Ohio
Kentucky


Strongest Sentiment by Region (Subset 6)
Sentiment
Minnesota
Wisconsin
Iowa
Missouri
Arkansas


Strongest Sentiment by Region (Subset 7)
Sentiment
Nebraska
Kansas
Oklahoma
Texas
Louisiana


Strongest Sentiment by Region (Subset 8)
Sentiment
Montana
North Dakota
South Dakota
Wyoming
Colorado


Strongest Sentiment by Region (Subset 9)
Sentiment
Idaho
Nevada
Utah
Arizona
New Mexico


Strongest Sentiment by Region (Subset 10)
Sentiment
Alaska
Washington
Oregon
California
Hawaii

In [ ]:
```
# Group states in subsets of 5
subset_size = 5
num_subsets = len(agg_us_scores_df) // subset_size

for i in range(num_subsets):
    subset_start = i * subset_size
    subset_end = (i + 1) * subset_size
    subset_data = agg_us_scores_df.iloc[subset_start:subset_end]

    # Create a bar plot for the subset of states
    plt.figure(figsize=(12, 8))
    subset_data.set_index('user_location')[sentiment_columns].plot(kind='bar',
    plt.title(f'Sentiment Scores by State (Subset {i+1})')
    plt.xlabel('State')
    plt.ylabel('Sentiment Score')
    plt.xticks(rotation=45)
    plt.legend(title='Sentiment')
    plt.tight_layout()

    plt.show()
```

In [ ]:

In [ ]: