In [3]:
```
import json
import pandas as pd
import matplotlib.pyplot as plt
```

In [4]:
```
df1 = pd.read_json('dutch_tweets_chunk1.json')
df2 = pd.read_json('dutch_tweets_chunk2.json')
df3 = pd.read_json('dutch_tweets_chunk3.json')
df4 = pd.read_json('dutch_tweets_chunk4.json')
df5 = pd.read_json('dutch_tweets_chunk5.json')
df6 = pd.read_json('dutch_tweets_chunk6.json')
df7 = pd.read_json('dutch_tweets_chunk7.json')
df8 = pd.read_json('dutch_tweets_chunk8.json')
df9 = pd.read_json('dutch_tweets_chunk9.json')
```

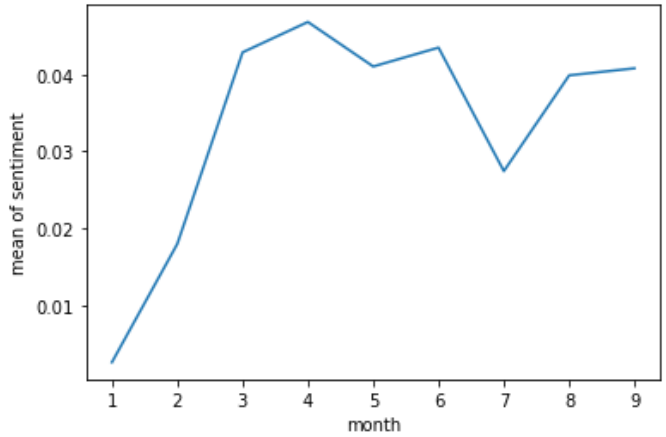
In [11]:
```
frames = [df1, df2, df3, df4, df5, df6, df7, df8, df9]
combineddf = pd.concat(frames)
```

In [12]:
```
meanOfSentiment = combineddf.groupby('month')['sentiment_pattern'].mean()
meanOfSentiment = meanOfSentiment.drop(meanOfSentiment.index[9]) # cleaning the last value which is 2020, the sentiment_pattern v
print(meanOfSentiment)
```

```
month
1.0       0.002562
2.0       0.017940
3.0       0.042862
4.0       0.046813
5.0       0.041024
6.0       0.043478
7.0       0.027415
8.0       0.039871
9.0       0.040785
2020.0    0.000000
Name: sentiment_pattern, dtype: float64
```

In [177]:
```
meanOfSentiment.plot(xlabel="month", ylabel="mean of sentiment")
```

Out[177]: <AxesSubplot:xlabel='month', ylabel='mean of sentiment'>



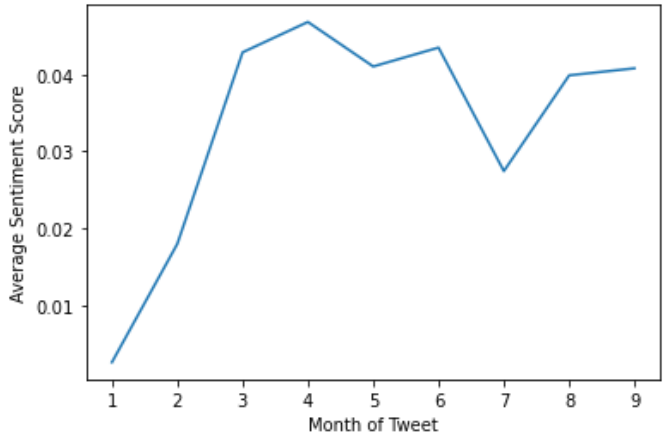
In [ ]:

In [150]:

```
0.0009361542782250515
```

In [39]:
```
meanOfSentiment.plot(xlabel='Month of Tweet', ylabel="Average Sentiment Score")
```

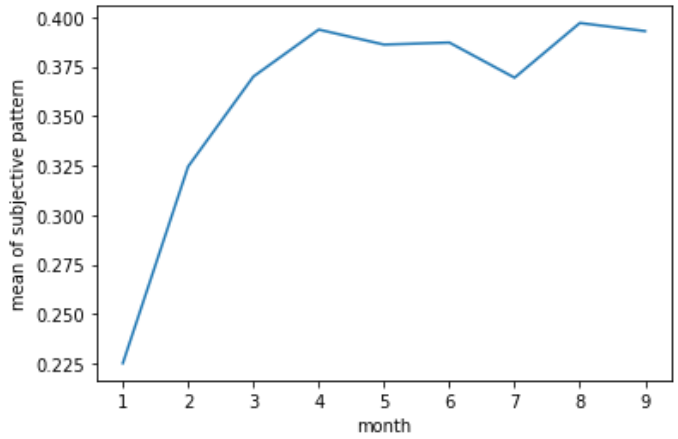
Out[39]: <AxesSubplot:xlabel='Month of Tweet', ylabel='Average Sentiment Score'>



In [40]:
```
meanOfSubjectivity = combineddf.groupby('month')['subjective_pattern'].mean()
list(meanOfSubjectivity)
print(meanOfSubjectivity)
meanOfSubjectivity = meanOfSubjectivity.drop(meanOfSubjectivity.index[9]) # cleaning the last value which is 2020, the sentiment_
meanOfSubjectivity.plot(xlabel="month", ylabel="mean of subjective pattern")
```

```
month
1.0       0.225120
2.0       0.324506
3.0       0.369991
4.0       0.393729
5.0       0.386129
6.0       0.387181
7.0       0.369410
8.0       0.397063
9.0       0.392950
2020.0    0.000000
Name: subjective_pattern, dtype: float64
```

Out[40]: <AxesSubplot:xlabel='month', ylabel='mean of subjective pattern'>



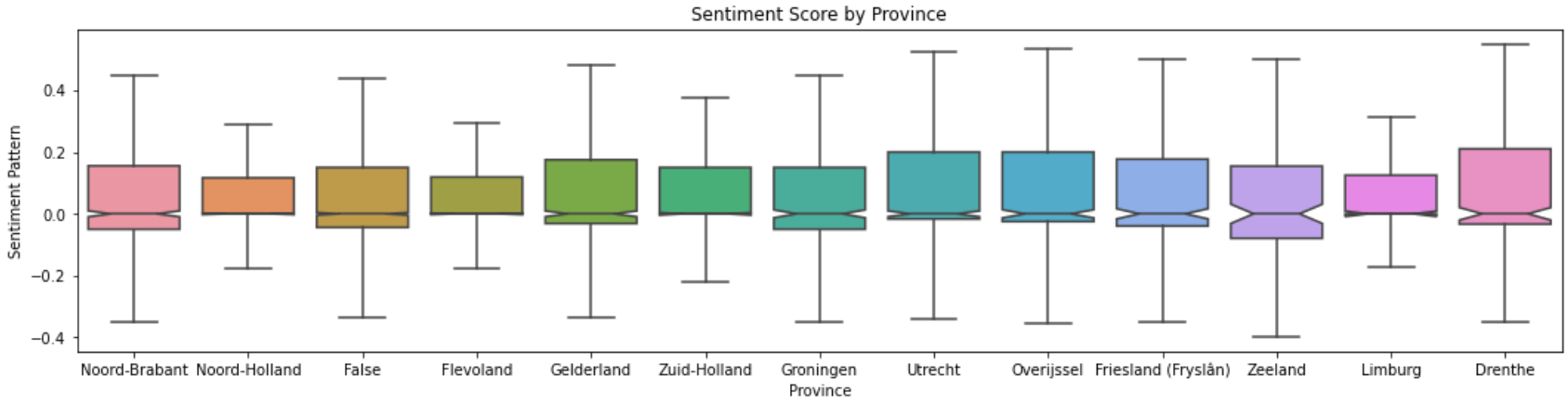
The graph above uses the sentiment score to understand how the score changes over the months. The graph clearly shows as time goes on, the average sentiment score increases. A potential reason for this is that there was probably more opinions about the longevity and potent of the COVID-19 virus as time went on. The longer the virus went on, the higher occurence of disagreements and polarization on the social media platform. Next, I want to examine whether the location within Netherlands has an effect on the sentiment score. Does urban vs non-urban matter in discrepancy between the sentiment scores?

In [42]:
```
locationdf = combineddf.groupby('province')['sentiment_pattern'].mean()
locationdf.describe()
print(locationdf)
```

```
province
False                  0.036801
Drenthe                0.049261
Flevoland              0.036178
Friesland (Fryslân)    0.033733
Gelderland             0.045287
Groningen              0.037555
Limburg                0.046575
Noord-Brabant          0.041627
Noord-Holland          0.035413
Overijssel             0.050964
Utrecht                0.058230
Zeeland                0.052609
Zuid-Holland           0.040630
Name: sentiment_pattern, dtype: float64
```

In [175]:
```
import seaborn as sns
plt.figure(figsize=(18,4))
plt.tight_layout()
sns.boxplot(x='province', y='sentiment_pattern', notch=True, data=combineddf, showfliers=False).set(title='Sentiment Score by Pro
plt.xlabel('Province')
plt.ylabel('Sentiment Pattern')
```

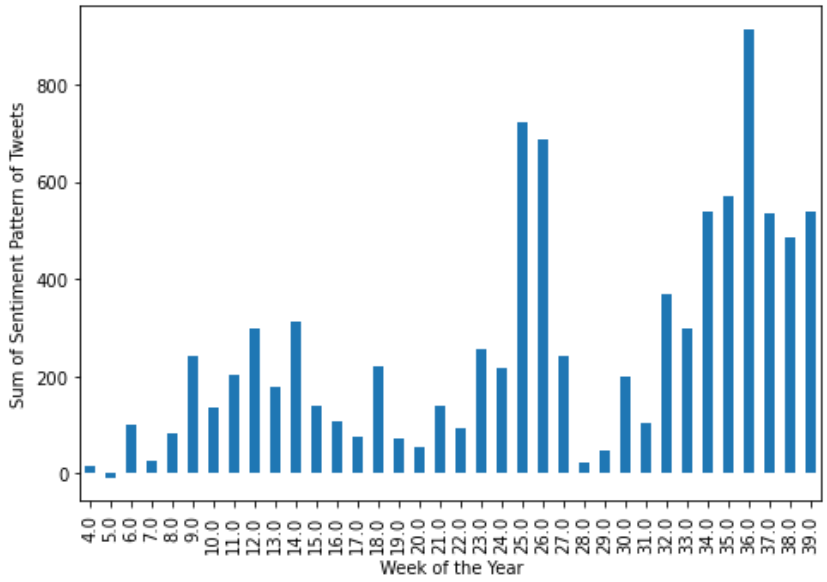
Out[175]: Text(0, 0.5, 'Sentiment Pattern')



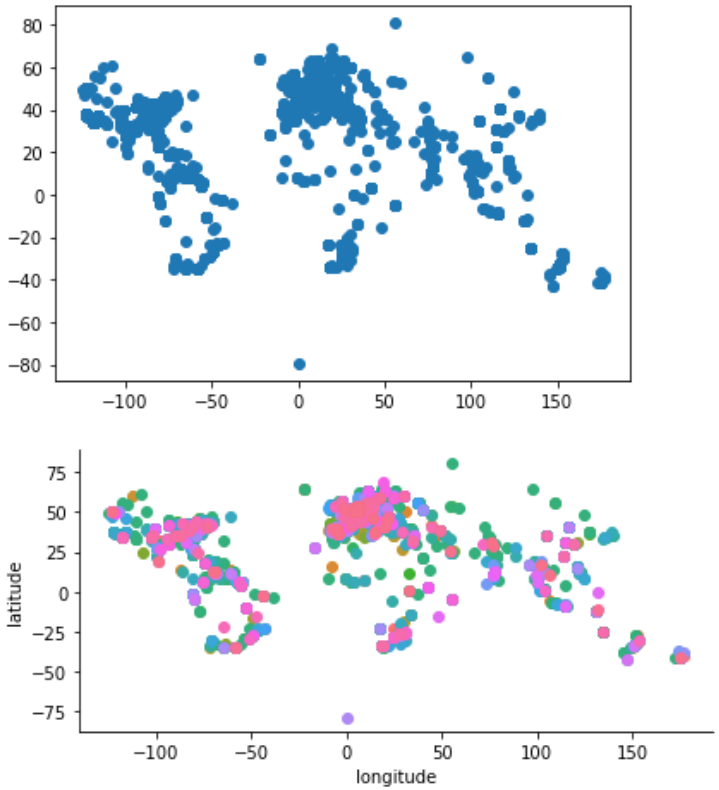
In [43]:
```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
weekofyeargroup = combineddf.groupby('weekofyear')['sentiment_pattern'].agg('sum')
#print(weekofyeargroup)

weekofyeargroup.plot.bar(xlabel="Week of the Year", ylabel="Sum of Sentiment Pattern of Tweets")
```

Out[43]: <Axes:xlabel='Week of the Year', ylabel='Sum of Sentiment Pattern of Tweets'>



In [209]:
```
import numpy as np
import numpy.random
import seaborn as sns
longitude = (np.asarray(combineddf['longitude']))
latitude = (np.asarray(combineddf['latitude']))
combineddf['long_lat'] = [', '.join(str(x) for x in y) for y in map(tuple, combineddf[['latitude', 'longitude']].values)]
plt.scatter(longitude, latitude)
fg = sns.FacetGrid(data=combineddf, hue='sentiment_pattern', aspect=2.0)
fg.map(plt.scatter, 'longitude', 'latitude')
plt.show()
```

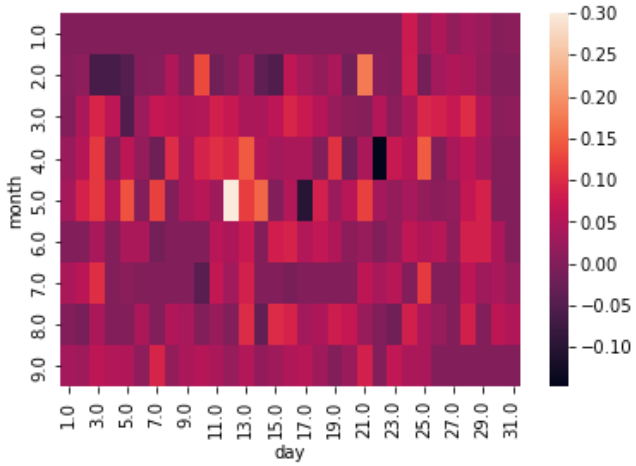


In [203]:
```
#result = combineddf.pivot(index='longitude', columns='latitude', values='sentiment_pattern')
with pd.option_context('mode.use_inf_as_null', True):
    combineddf = combineddf.dropna()
result = combineddf.reset_index().pivot_table(values='sentiment_pattern', index='day', columns='month', fill_value=0)

#result.set_index(combined)
```

In [186]:
```
import seaborn as sns

ax=sns.heatmap(result)
```



In [198]:
```
from sklearn.model_selection import train_test_split
import numpy as np
with pd.option_context('mode.use_inf_as_null', True):
    combineddf = combineddf.dropna()
y = combineddf['sentiment_pattern'].to_numpy()
X = combineddf[['day']].to_numpy()

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2, random_state=4)
```

In [199]:
```
np.isnan(X_train).any(), np.isnan(X_test).any()
```

Out[199]: (False, False)

In [200]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test = label_encoder.fit_transform(y_test)
```

In [201]:
```
import sklearn
from sklearn import neighbors
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV

base_clf = neighbors.KNeighborsClassifier()
parameters = {'n_neighbors': [1, 2, 5, 10, 15, 25], 'weights': ['uniform', 'distance']}

clf = GridSearchCV(base_clf, parameters, cv=3)
clf.fit(X_train, y_train)
print('Best Hyperparameters: ', clf.best_params_, '\n')

pred = clf.predict(X_test)
scores = clf.predict_proba(X_test)[:,1]

print('Accuracy: ', accuracy_score(y_test, pred))
#print('AUROC: ', roc_auc_score(y_test, scores))
```

```
C:\Users\sunay\anaconda3\lib\site-packages\sklearn\model_selection\_split.py:666: UserWarning: The least populated class in y ha
s only 1 members, which is less than n_splits=3.
  warnings.warn(("The least populated class in y has only %d"

Best Hyperparameters:  {'n_neighbors': 25, 'weights': 'uniform'}

Accuracy:  0.0009361542782250515
```

In [ ]:
```
```