FIN3210 Week 4 Assignment

Ma Kexuan

October 14, 2023

[1]:
```
import pandas as pd
import numpy as np
from pytrends.request import TrendReq
import matplotlib.pyplot as plt
from matplotlib.pyplot import MultipleLocator
import time
import warnings
warnings.filterwarnings("ignore")
import statsmodels.api as sm
```

[2]:
```
def extract_trends(wordlist, location):
    """
    Extract the google trends data of the given word list
    :param wordlist: list of words
    :return: pandas dataframe
    """
    count = 0
    for i in range(0, len(wordlist), 5):
        pytrend = TrendReq()
        pytrend.build_payload(kw_list=wordlist[i:i+5], timeframe='2018-9-1␣
 ↪2023-8-31', geo = location)
        py_res = pytrend.interest_over_time().reset_index()
        if count == 0:
            py_res.drop(columns=['isPartial'], axis=1, inplace=True)
            res = py_res
        else:
            py_res.drop(columns=['isPartial', 'date'], axis=1, inplace=True)
            res = pd.concat([res, py_res], axis=1)
        count += 1
        time.sleep(120)
    return res
```

0.1 Q1. Using Google Trends (https://trends.google.com/trends/?geo=US), construct a weekly index to capture political relations between U.S. and China from the US perspective, draw the variable in a graph, and discuss its time-series variation

[3]:
```
us_china_keywords = [
    "Tariffs",
    "South China Sea",
    "Huawei",
    "Trade War",
    "Made in China",
    "Tibet",
    "Hong Kong",
    "Taiwan",
    "U.S.-China",
    "5G"
]
```

[4]:
```
# Use to extract data from Google Trend
# res = extract_trends(us_china_keywords, 'US')
# res
```

[5]:
```
origin_index = pd.read_csv('us_china.csv')
# Drop it since it contains no information according to the data
origin_index.drop(['U.S.-China'], axis=1, inplace=True)
index_list = origin_index.columns.tolist()[1:]
# Construct the index
origin_index['total_index'] = origin_index[index_list].mean(axis=1)
origin_index
```

[5]:

	date	Tariffs	South China Sea	Huawei	Trade War	Made in China	\
0	2018-09-02	11	2	17	7	5	
1	2018-09-09	11	2	19	8	5	
2	2018-09-16	25	2	17	12	5	
3	2018-09-23	15	2	19	10	5	
4	2018-09-30	10	4	16	7	5	
..	…	…	…	…	…	…	
256	2023-07-30	1	1	6	1	5	
257	2023-08-06	1	1	7	1	5	
258	2023-08-13	2	1	6	2	5	
259	2023-08-20	2	1	6	2	5	
260	2023-08-27	3	2	9	3	6	

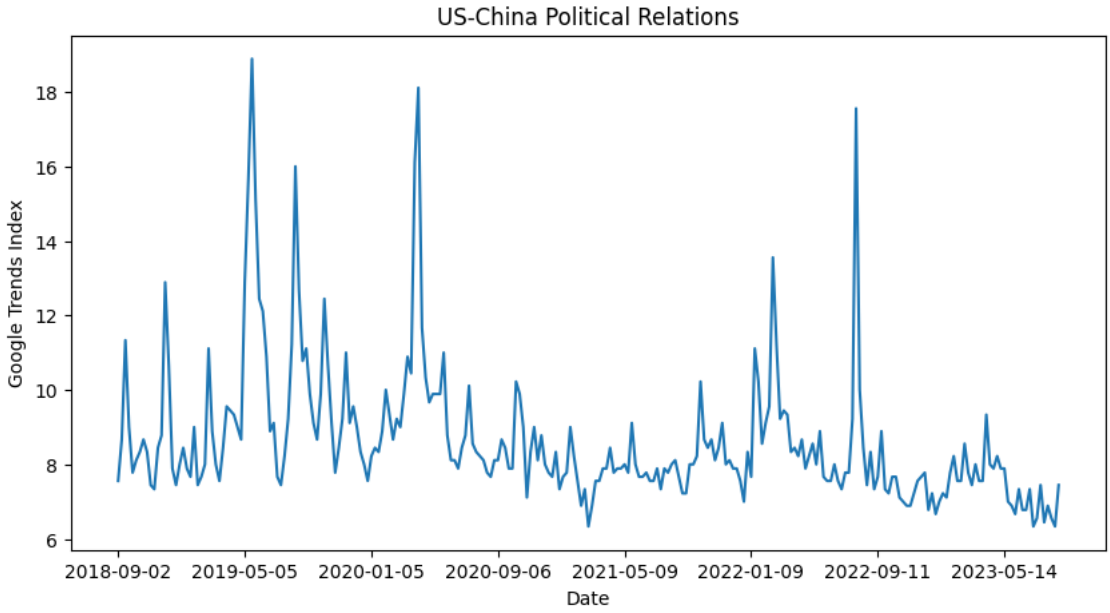
	Tibet	Hong Kong	Taiwan	5G	total_index
0	1	16	6	3	7.555556
1	1	18	7	7	8.666667
2	1	24	9	7	11.333333
3	1	17	7	5	9.000000

```
4        1         15       6   6     7.777778
..     …         …      …   ..          …
256      1         14      10  19     6.444444
257      1         14      11  21     6.888889
258      1         13      10  19     6.555556
259      1         12      10  18     6.333333
260      1         14      10  19     7.444444

[261 rows x 11 columns]
```

[6]:
```
corr_data = origin_index.drop(columns=['date','total_index'], axis=1)
corr_data.corr()
```

[6]:
```
                  Tariffs  South China Sea    Huawei  Trade War  \
Tariffs          1.000000         0.064757  0.508916   0.887880
South China Sea  0.064757         1.000000  0.031378   0.046260
Huawei           0.508916         0.031378  1.000000   0.528077
Trade War        0.887880         0.046260  0.528077   1.000000
Made in China   -0.133332         0.167426 -0.212102  -0.156281
Tibet            0.059853         0.068504  0.045272   0.088927
Hong Kong        0.293850        -0.009325  0.220016   0.408141
Taiwan          -0.151932         0.073614 -0.214053  -0.150847
5G              -0.372672        -0.024228 -0.350924  -0.433250

                 Made in China     Tibet  Hong Kong    Taiwan        5G
Tariffs              -0.133332  0.059853   0.293850 -0.151932 -0.372672
South China Sea       0.167426  0.068504  -0.009325  0.073614 -0.024228
Huawei               -0.212102  0.045272   0.220016 -0.214053 -0.350924
Trade War            -0.156281  0.088927   0.408141 -0.150847 -0.433250
Made in China         1.000000  0.095936  -0.071522  0.113407  0.405090
Tibet                 0.095936  1.000000  -0.080955  0.191257 -0.056895
Hong Kong            -0.071522 -0.080955   1.000000 -0.097992 -0.289373
Taiwan                0.113407  0.191257  -0.097992  1.000000  0.196832
5G                    0.405090 -0.056895  -0.289373  0.196832  1.000000
```

[7]:
```
plt.figure(figsize=(10, 5))
plt.plot(origin_index['date'], origin_index['total_index'])
plt.title('US-China Political Relations')
plt.xlabel('Date')
plt.ylabel('Google Trends Index')
x_major_locator=MultipleLocator(35)
ax = plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
plt.show()
```



0.2 Q2. Using Google Trends (https://trends.google.com/trends/?geo=US) or Baidu Index (http://index.baidu.com/), construct an index to capture investor sentiment in the Chinese market, draw the variable in a graph, and discuss its time-series variation.

[8]:
```
positive_words_list = ['boom', 'buy', 'credit', 'gain', 'profit',
                       'reward', 'surge', 'rise', 'boost', 'win']
negative_words_list = ['bankrupt', 'capital', 'decline', 'default', 'fall',
                       'inflation', 'liability', 'loss', 'recession', 'short']
```

[9]:
```
# Use to extract data from Google Trend
# pos_result = extract_trends(positive_words_list, 'CN')
# neg_result = extract_trends(negative_words_list, 'CN')
```

[10]:
```
pos_result = pd.read_csv('positive_words.csv')
neg_result = pd.read_csv('negative_words.csv')
```

[11]:
```
pos_result[positive_words_list].corr()
```

[11]:

	boom	buy	credit	gain	profit	reward	surge
boom	1.000000	-0.229926	-0.112970	-0.003785	0.078841	0.060229	-0.248064
buy	-0.229926	1.000000	0.459562	-0.054518	-0.241693	-0.099755	0.111902
credit	-0.112970	0.459562	1.000000	-0.024404	-0.070394	-0.018074	0.114787
gain	-0.003785	-0.054518	-0.024404	1.000000	0.135644	-0.002035	-0.010019
profit	0.078841	-0.241693	-0.070394	0.135644	1.000000	0.137452	-0.014305
reward	0.060229	-0.099755	-0.018074	-0.002035	0.137452	1.000000	-0.011389
surge	-0.248064	0.111902	0.114787	-0.010019	-0.014305	-0.011389	1.000000

```
rise    0.002136 -0.042888 -0.079913  0.098343  0.126010  0.030958 -0.156459
boost  -0.093268  0.314673  0.274375 -0.112923 -0.117124 -0.088233 -0.053258
win     0.205413 -0.158791 -0.189805  0.053169  0.105871  0.077464 -0.073554

            rise     boost       win
boom    0.002136 -0.093268  0.205413
buy    -0.042888  0.314673 -0.158791
credit -0.079913  0.274375 -0.189805
gain    0.098343 -0.112923  0.053169
profit  0.126010 -0.117124  0.105871
reward  0.030958 -0.088233  0.077464
surge  -0.156459 -0.053258 -0.073554
rise    1.000000 -0.158276 -0.006620
boost  -0.158276  1.000000 -0.078053
win    -0.006620 -0.078053  1.000000
```

[12]:
```
neg_result[negative_words_list].corr()
```

[12]:
```
           bankrupt   capital   decline   default      fall  inflation  \
bankrupt   1.000000  0.172598 -0.074039  0.016845  0.036303  -0.000916
capital    0.172598  1.000000  0.024463 -0.042862  0.079861  -0.062564
decline   -0.074039  0.024463  1.000000  0.029648 -0.040891  -0.058599
default    0.016845 -0.042862  0.029648  1.000000 -0.004481   0.009918
fall       0.036303  0.079861 -0.040891 -0.004481  1.000000  -0.000501
inflation -0.000916 -0.062564 -0.058599  0.009918 -0.000501   1.000000
liability  0.087584  0.056222 -0.022007 -0.051609  0.015856  -0.057728
loss       0.076411  0.100815  0.011757  0.265133 -0.067665   0.003452
recession -0.116179 -0.093051  0.211840 -0.130195 -0.006859   0.169077
short      0.194869  0.146783 -0.029136  0.098861 -0.026079   0.192370

           liability      loss  recession     short
bankrupt    0.087584  0.076411  -0.116179  0.194869
capital     0.056222  0.100815  -0.093051  0.146783
decline    -0.022007  0.011757   0.211840 -0.029136
default    -0.051609  0.265133  -0.130195  0.098861
fall        0.015856 -0.067665  -0.006859 -0.026079
inflation  -0.057728  0.003452   0.169077  0.192370
liability   1.000000  0.004283   0.063456  0.048090
loss        0.004283  1.000000   0.006106  0.098029
recession   0.063456  0.006106   1.000000 -0.041244
short       0.048090  0.098029  -0.041244  1.000000
```

[13]:
```
pos_result['pos_total_index'] = pos_result[positive_words_list].mean(axis=1)
neg_result['neg_total_index'] = neg_result[negative_words_list].mean(axis=1)
neg_result.drop(['date'], axis=1, inplace=True)
result = pd.concat([pos_result, neg_result], axis=1)
# Calculate the overall sentiment index based on pos&neg
```

```
result['total_index'] = result['pos_total_index'] - result['neg_total_index']
result['y_w'] = pd.to_datetime(result['date']).dt.strftime('%Y-%U')
result
```

[13]:

```
           date  boom  buy  credit  gain  profit  reward  surge  rise  boost  \
0    2018-09-02     4   45      26     3       5       0      8    12     39
1    2018-09-09     4   55      32     5       9       3     11    11     42
2    2018-09-16     7   55      34     9      10       6      5    18     35
3    2018-09-23     3   58      31     9       5       4     16    22     30
4    2018-09-30     0   71      39     6       8       7     16     9     32
..          …   …  …     …   …     …      …     …   …     …
256  2023-07-30     4   48      31     7       6       8     16    16     21
257  2023-08-06     3   42      26     9       8       5     22    18     24
258  2023-08-13     0   43      36    11       9      11     15    12     27
259  2023-08-20     4   42      25     6       5       6     10    18     18
260  2023-08-27     4   51      20     6       4       7     16    13     21

     …  default  fall  inflation  liability  loss  recession  short  \
0    …       78    17          0          0    54          4     74
1    …       77    29         10          0    55          0     58
2    …       69    26         15         12    49          0     57
3    …       84    41          8          6    57          0     66
4    …       37    43         14          0    48          0     75
..   …      …   …        …         …   …        …    …
256  …       63    27         14          8    55          5     45
257  …       59    21          6          8    52          3     55
258  …       61    33         12          4    63          5     60
259  …       57    27          5          8    50          6     58
260  …       45    15          5          3    56          3     64

     neg_total_index  total_index      y_w
0               28.7         -9.2  2018-35
1               27.6         -5.3  2018-36
2               27.8         -3.3  2018-37
3               31.2         -8.9  2018-38
4               26.7         -3.3  2018-39
..               …          …       …
256             27.6         -6.9  2023-31
257             24.9         -5.5  2023-32
258             30.2         -8.6  2023-33
259             26.4         -7.7  2023-34
260             25.4         -6.3  2023-35

[261 rows x 25 columns]
```

[14]:

```
ret_data = pd.read_csv('return.csv')
# Calculate the weekly return of Shanghai market index
```

```
ret_data['Idxtrd08'] = ret_data['Idxtrd08']/100
ret_data['year_week'] = pd.to_datetime(ret_data['Idxtrd01']).dt.
 ↪strftime('%Y-%U')
ret_data['cum_ret'] = 1 + ret_data['Idxtrd08']
ret_idx = ret_data.groupby('year_week').agg({'cum_ret': np.prod}).reset_index()
ret_idx['week_ret'] = ret_idx['cum_ret'] - 1
ret_idx
```

[14]:

	year_week	cum_ret	week_ret
0	2018-35	0.991580	-0.008420
1	2018-36	0.992355	-0.007645
2	2018-37	1.043199	0.043199
3	2018-38	1.008531	0.008531
4	2018-40	0.923995	-0.076005
..	…	…	…
251	2023-31	1.003711	0.003711
252	2023-32	0.969941	-0.030059
253	2023-33	0.982035	-0.017965
254	2023-34	0.978328	-0.021672
255	2023-35	1.018212	0.018212

[256 rows x 3 columns]

[15]:
```
reg_data = pd.merge(result, ret_idx, left_on='y_w', right_on='year_week',␣
 ↪how='left')
reg_data = reg_data[['year_week', 'total_index', 'week_ret']]
reg_data.dropna(inplace=True)
reg_data
```

[15]:

	year_week	total_index	week_ret
0	2018-35	-9.2	-0.008420
1	2018-36	-5.3	-0.007645
2	2018-37	-3.3	0.043199
3	2018-38	-8.9	0.008531
5	2018-40	-6.4	-0.076005
..	…	…	…
256	2023-31	-6.9	0.003711
257	2023-32	-5.5	-0.030059
258	2023-33	-8.6	-0.017965
259	2023-34	-7.7	-0.021672
260	2023-35	-6.3	0.018212

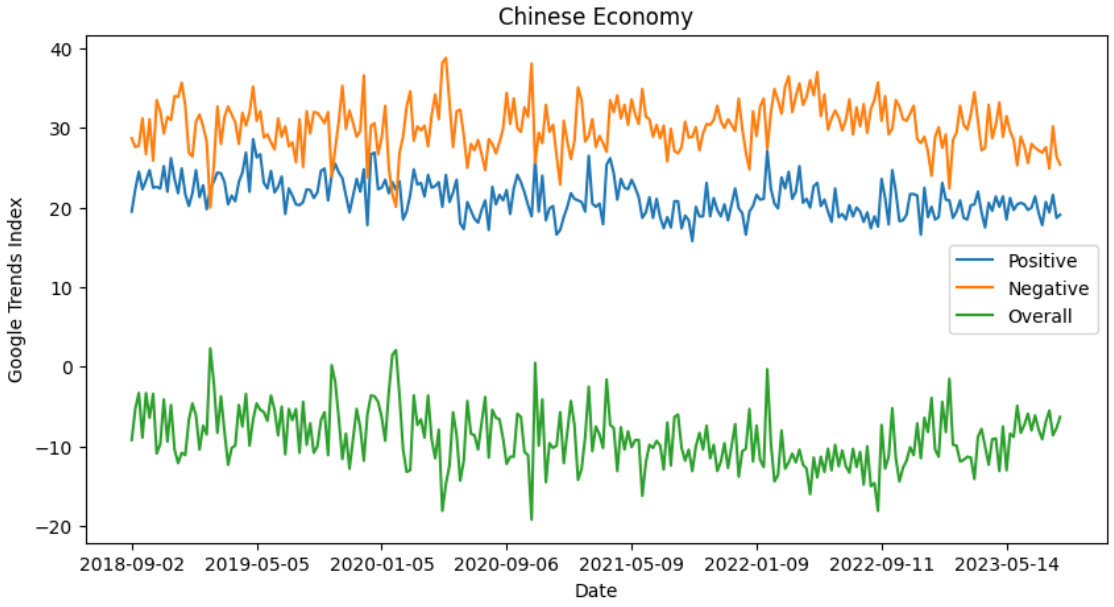
[254 rows x 3 columns]

[16]:
```
# Perform the OLS Regression
X = reg_data['total_index']
y = reg_data['week_ret']
```

```
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
model.summary()
```

[16]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:               week_ret   R-squared:                       0.002
Model:                           OLS   Adj. R-squared:                 -0.002
Method:                Least Squares   F-statistic:                    0.5385
Date:               Sat, 14 Oct 2023   Prob (F-statistic):              0.464
Time:                       18:18:13   Log-Likelihood:                601.44
No. Observations:                254   AIC:                            -1199.
Df Residuals:                    252   BIC:                            -1192.
Df Model:                          1
Covariance Type:           nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0035      0.004      0.862      0.389      -0.005       0.012
total_index    0.0003      0.000      0.734      0.464      -0.001       0.001
==============================================================================
Omnibus:                        4.268   Durbin-Watson:                   2.123
Prob(Omnibus):                  0.118   Jarque-Bera (JB):                4.990
Skew:                          -0.118   Prob(JB):                       0.0825
Kurtosis:                       3.645   Cond. No.                         28.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[17]:
```
plt.figure(figsize=(10, 5))
plt.plot(result['date'], result['pos_total_index'], label='Positive')
plt.plot(result['date'], result['neg_total_index'], label='Negative')
plt.plot(result['date'], result['total_index'], label='Overall')
plt.title('Chinese Economy')
plt.xlabel('Date')
plt.ylabel('Google Trends Index')
plt.legend()
x_major_locator=MultipleLocator(35)
ax = plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
plt.show()
```


Chinese Economy
Google Trends Index
40
30
20
10
0
−10
−20
2018-09-02
2019-05-05
2020-01-05
2020-09-06
2021-05-09
2022-01-09
2022-09-11
2023-05-14
Date
Positive
Negative
Overall