GWP 3 Submission

Group No - 846

- Primary Theme - Statistical Related Risk: Volatility & Statistical Related Risk: Correlation
- Secondary Theme - Fallout: Model Failure & Crises

Section - 1 (Theoretical Set up)

1. Statistical Related Risk: Volatility

Volatility, also called Standard Deviation, is a measure of dispersion of data points with respect to its mean. It is square root of Variance.

$$\sigma^2(X) = \frac{\Sigma(X - E(X))^2}{n-1}$$

$$\sigma(X) = \sqrt{(\frac{\Sigma(X - E(X))^2}{n-1}))}$$

Covariance is defined as measure of whether two variables move inline with each other or Anti to each other. It can range from minus infinity to plus infinity.

$$Covar(X,Y) = \frac{\Sigma(X - E(X)) * (Y - E(Y))}{n-1}$$

1. Statistical Related Risk: Correlation

Correlation is defined as the strength of linear relationship between two variables whith an assumption that both variables have a linear relationship. It ranges from +1 to -1. The value of 0 corresponds to no correlation.

$$\rho = \frac{Covar(X,Y)}{\sigma(X) * \sigma(Y)}$$

As an example for both Correlation and Volatility, we will illustrate Fama French 3 Factor model.

Fama French 3 Factor model tries to model returns based on three factors:

- Benchmark excess returns over the risk free rate
- Size premium
- Value premium

$$r_{it} - r_{ft} = \alpha + \beta_1 * (r_{mt} - f_{ft}) + \beta_2 * SMB_t + \beta_3 * HML_t + \epsilon_{it}$$

Here,

- SMB is Size premium
- HML is value premium

Section 2 - Illustrations

1. Fama French 3 Factor Model

In [2]:
```
#import necessary packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
import scipy.stats as s

%matplotlib inline
```

1. Data Importing and Cleaning

In [4]:
```
#1st dataset
ind_port=pd.read_csv('./Industry_Portfolios.csv')

ind_port.head(20)
```

Out[4]:

	This file was created by CMPT_IND_RETS using the 202205 CRSP database.	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8
0	It contains value- and equal-weighted returns ...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	The portfolios are constructed at the end of J...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	The annual returns are from January to December.	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	Missing data are indicated by -99.99 or -999.	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	Average Value Weighted Returns -- Monthly	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10	NaN	Agric	Food	Soda	Beer	Smoke	Toys	Fun	Books
11	192607	2.37	0.12	-99.99	-5.19	1.29	8.65	2.5	50.21
12	192608	2.23	2.68	-99.99	27.03	6.5	16.81	-0.76	42.98
13	192609	-0.57	1.58	-99.99	4.02	1.26	8.33	6.42	-4.91
14	192610	-0.46	-3.68	-99.99	-3.31	1.06	-1.4	-5.09	5.37
15	192611	6.75	6.26	-99.99	7.29	4.55	0	1.82	-6.4
16	192612	-3.27	0.18	-99.99	-4.09	2.55	2.48	2.14	-3.29
17	192701	-3.66	-0.16	-99.99	0.57	-0.35	1.73	1.88	1.21
18	192702	7.65	3.66	-99.99	12.83	1.49	-6.12	2.43	10.31
19	192703	-0.6	2.74	-99.99	-13.56	5.51	-8.89	1.93	-7.83

20 rows × 50 columns

In [5]:
```
#Extracting the value weighted returns monthly data
value_wted_returns=ind_port.iloc[11:1162,:]
value_wted_returns.columns=['Time', 'Agric', 'Food ', 'Soda ', 'Beer ', 'Smoke', 'Toys '
       'Books', 'Hshld', 'Clths', 'Hlth ', 'MedEq', 'Drugs', 'Chems', 'Rubbr',
       'Txtls', 'BldMt', 'Cnstr', 'Steel', 'FabPr', 'Mach ', 'ElcEq', 'Autos',
       'Aero ', 'Ships', 'Guns ', 'Gold ', 'Mines', 'Coal ', 'Oil  ', 'Util ',
       'Telcm', 'PerSv', 'BusSv', 'Hardw', 'Softw', 'Chips', 'LabEq', 'Paper',
       'Boxes', 'Trans', 'Whlsl', 'Rtail', 'Meals', 'Banks', 'Insur', 'RlEst',
       'Fin  ', 'Other']
value_wted_returns.reset_index(inplace=True)
value_wted_returns.drop(columns=['index'],inplace=True)
value_wted_returns.head()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_12824\425753230.py:11: SettingWithCopyWarnin
g:
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
  value_wted_returns.drop(columns=['index'],inplace=True)
```

Out[5]:

	Time	Agric	Food	Soda	Beer	Smoke	Toys	Fun	Books	Hshld	...	Boxes	Trans	Whlsl	Rtail	Meals
0	192607	2.37	0.12	-99.99	-5.19	1.29	8.65	2.5	50.21	-0.48	...	7.7	1.92	-23.79	0.07	1.87
1	192608	2.23	2.68	-99.99	27.03	6.5	16.81	-0.76	42.98	-3.58	...	-2.38	4.85	5.39	-0.75	-0.13
2	192609	-0.57	1.58	-99.99	4.02	1.26	8.33	6.42	-4.91	0.73	...	-5.54	0.08	-7.87	0.25	-0.56
3	192610	-0.46	-3.68	-99.99	-3.31	1.06	-1.4	-5.09	5.37	-4.68	...	-5.08	-2.62	-15.38	-2.2	-4.11
4	192611	6.75	6.26	-99.99	7.29	4.55	0	1.82	-6.4	-0.54	...	3.84	1.61	4.67	6.52	4.33

5 rows × 50 columns

In [6]:
```
value_wted_returns.shape
```

```
#This shape corresponds to similar data in the csv file
```

Out[6]:

```
(1151, 50)
```

In [7]:

```
#Importing and Extractying the required Fama-French Data
ff_data=pd.read_csv('./F-F_Research_Data_5_Factors_2x3.csv')
ff_data.head(20)
```

Out[7]:

	This file was created by CMPT_ME_BEME_OP_INV_RETS using the 202205 CRSP database.	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6
0	The 1-month TBill return is from Ibbotson and ...	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	Mkt-RF	SMB	HML	RMW	CMA	RF
3	196307	-0.39	-0.44	-0.89	0.68	-1.23	0.27
4	196308	5.07	-0.75	1.68	0.36	-0.34	0.25
5	196309	-1.57	-0.55	0.08	-0.71	0.29	0.27
6	196310	2.53	-1.37	-0.14	2.8	-2.02	0.29
7	196311	-0.85	-0.89	1.81	-0.51	2.31	0.27
8	196312	1.83	-2.07	-0.08	0.03	-0.04	0.29
9	196401	2.24	0.11	1.47	0.17	1.51	0.3
10	196402	1.54	0.3	2.74	-0.05	0.9	0.26
11	196403	1.41	1.36	3.36	-2.21	3.19	0.31
12	196404	0.1	-1.59	-0.58	-1.27	-1.04	0.29
13	196405	1.42	-0.64	1.82	-0.16	0.14	0.26
14	196406	1.27	0.31	0.63	-0.28	-0.15	0.3
15	196407	1.74	0.47	0.75	0.04	1.94	0.3
16	196408	-1.44	0.42	0.08	0.15	0.33	0.28
17	196409	2.69	-0.33	1.7	-0.54	0.61	0.28
18	196410	0.59	0.91	1.17	-0.38	0.43	0.29
19	196411	0	-0.15	-1.96	0.62	-0.26	0.29

In [8]:

```
ff_monthly_data=ff_data.iloc[3:710,:]
ff_monthly_data.columns=['Time', 'Mkt-RF', 'SMB', 'HML', 'RMW', 'CMA', 'RF']
ff_monthly_data.head()
ff_monthly_data.reset_index(inplace=True)
ff_monthly_data.drop(columns=['index'],inplace=True)
ff_monthly_data.head()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_12824\3190668661.py:5: SettingWithCopyWarnin
g:
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
  ff_monthly_data.drop(columns=['index'],inplace=True)
```

Out[8]:

	Time	Mkt-RF	SMB	HML	RMW	CMA	RF

0	196307	-0.39	-0.44	-0.89	0.68	-1.23	0.27
1	196308	5.07	-0.75	1.68	0.36	-0.34	0.25
2	196309	-1.57	-0.55	0.08	-0.71	0.29	0.27
3	196310	2.53	-1.37	-0.14	2.8	-2.02	0.29
4	196311	-0.85	-0.89	1.81	-0.51	2.31	0.27

In [9]:
```
ff_monthly_data.shape
```
Out[9]:
```
(707, 7)
```

1. Preprocessing of data

For returns dataset

- For industry portfolio data, it is explicitly given that the value of -99.99 or -999 correspond to missing data. So, in order to improve data quality we impute values(mean) these columns(industries).

In [11]:
```
#proportion of missing data
mask=(value_wted_returns== '-99.99') | (value_wted_returns== '-999')
mask.mean()
```
Out[11]:
```
Time     0.000000
Agric    0.000000
Food     0.000000
Soda     0.385752
Beer     0.000000
Smoke    0.000000
Toys     0.000000
Fun      0.000000
Books    0.000000
Hshld    0.000000
Clths    0.000000
Hlth     0.448306
MedEq    0.000000
Drugs    0.000000
Chems    0.000000
Rubbr    0.052129
Txtls    0.000000
BldMt    0.000000
Cnstr    0.000000
Steel    0.000000
FabPr    0.385752
Mach     0.000000
ElcEq    0.000000
Autos    0.000000
Aero     0.000000
Ships    0.000000
Guns     0.385752
Gold     0.385752
Mines    0.000000
Coal     0.000000
Oil      0.000000
Util     0.000000
Telcm    0.000000
PerSv    0.010426
BusSv    0.000000
Hardw    0.000000
Softw    0.406603
Chips    0.000000
```

```
LabEq    0.000000
Paper    0.031277
Boxes    0.000000
Trans    0.000000
Whlsl    0.000000
Rtail    0.000000
Meals    0.000000
Banks    0.000000
Insur    0.000000
RlEst    0.000000
Fin      0.000000
Other    0.000000
dtype: float64
```

In [12]:

```
impute_col=['Rubbr','PerSv','Paper','Soda ', 'Hlth ', 'FabPr', 'Guns ', 'Gold ','Softw']

value_wted_returns[value_wted_returns.columns[1:]]=value_wted_returns[value_wted_returns
value_wted_returns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1151 entries, 0 to 1150
Data columns (total 50 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Time    1151 non-null   object 
 1   Agric   1151 non-null   float64
 2   Food    1151 non-null   float64
 3   Soda    1151 non-null   float64
 4   Beer    1151 non-null   float64
 5   Smoke   1151 non-null   float64
 6   Toys    1151 non-null   float64
 7   Fun     1151 non-null   float64
 8   Books   1151 non-null   float64
 9   Hshld   1151 non-null   float64
 10  Clths   1151 non-null   float64
 11  Hlth    1151 non-null   float64
 12  MedEq   1151 non-null   float64
 13  Drugs   1151 non-null   float64
 14  Chems   1151 non-null   float64
 15  Rubbr   1151 non-null   float64
 16  Txtls   1151 non-null   float64
 17  BldMt   1151 non-null   float64
 18  Cnstr   1151 non-null   float64
 19  Steel   1151 non-null   float64
 20  FabPr   1151 non-null   float64
 21  Mach    1151 non-null   float64
 22  ElcEq   1151 non-null   float64
 23  Autos   1151 non-null   float64
 24  Aero    1151 non-null   float64
 25  Ships   1151 non-null   float64
 26  Guns    1151 non-null   float64
 27  Gold    1151 non-null   float64
 28  Mines   1151 non-null   float64
 29  Coal    1151 non-null   float64
 30  Oil     1151 non-null   float64
 31  Util    1151 non-null   float64
 32  Telcm   1151 non-null   float64
 33  PerSv   1151 non-null   float64
 34  BusSv   1151 non-null   float64
 35  Hardw   1151 non-null   float64
 36  Softw   1151 non-null   float64
 37  Chips   1151 non-null   float64
 38  LabEq   1151 non-null   float64
 39  Paper   1151 non-null   float64
 40  Boxes   1151 non-null   float64
 41  Trans   1151 non-null   float64
```

```
 42  Whlsl   1151 non-null   float64
 43  Rtail   1151 non-null   float64
 44  Meals   1151 non-null   float64
 45  Banks   1151 non-null   float64
 46  Insur   1151 non-null   float64
 47  RlEst   1151 non-null   float64
 48  Fin     1151 non-null   float64
 49  Other   1151 non-null   float64
dtypes: float64(49), object(1)
memory usage: 449.7+ KB
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_12824\1382093669.py:3: SettingWithCopyWarnin
g:
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
  value_wted_returns[value_wted_returns.columns[1:]]=value_wted_returns[value_wted_retur
ns.columns[1:]].astype(float)
```

In [13]:

```
#impute with mean values
mask=(value_wted_returns!=-99.99)&(value_wted_returns!=-999)
impute_values=value_wted_returns[impute_col][mask].dropna().mean()
impute_values
```

Out[13]:

```
Rubbr    1.043937
PerSv    0.621921
Paper    0.904520
Soda     1.124394
Hlth     1.010283
FabPr    0.830693
Guns     1.284961
Gold     0.925449
Softw    1.041669
dtype: float64
```

In [14]:

```
#impute values
for i in impute_col:
    value_wted_returns[i]=np.where(((value_wted_returns[i]==-99.99)|(value_wted_returns[
                                   impute_values[i],value_wted_returns[i])
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_12824\2214042178.py:3: SettingWithCopyWarnin
g:
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
  value_wted_returns[i]=np.where(((value_wted_returns[i]==-99.99)|(value_wted_returns['R
ubbr']==-999)),
```

- For F-F dataset, we don't need any particular preprocessing except converting the data type to float(except the 'Time' column).
- Furthermore, we do not disturb the 'Time' column format as they are same across all dataframes.

1. Analysis

In [16]:

```
#Industry wise average returns
ri_bar=value_wted_returns.drop(columns=['Time']).mean().sort_values()
ri_bar
```

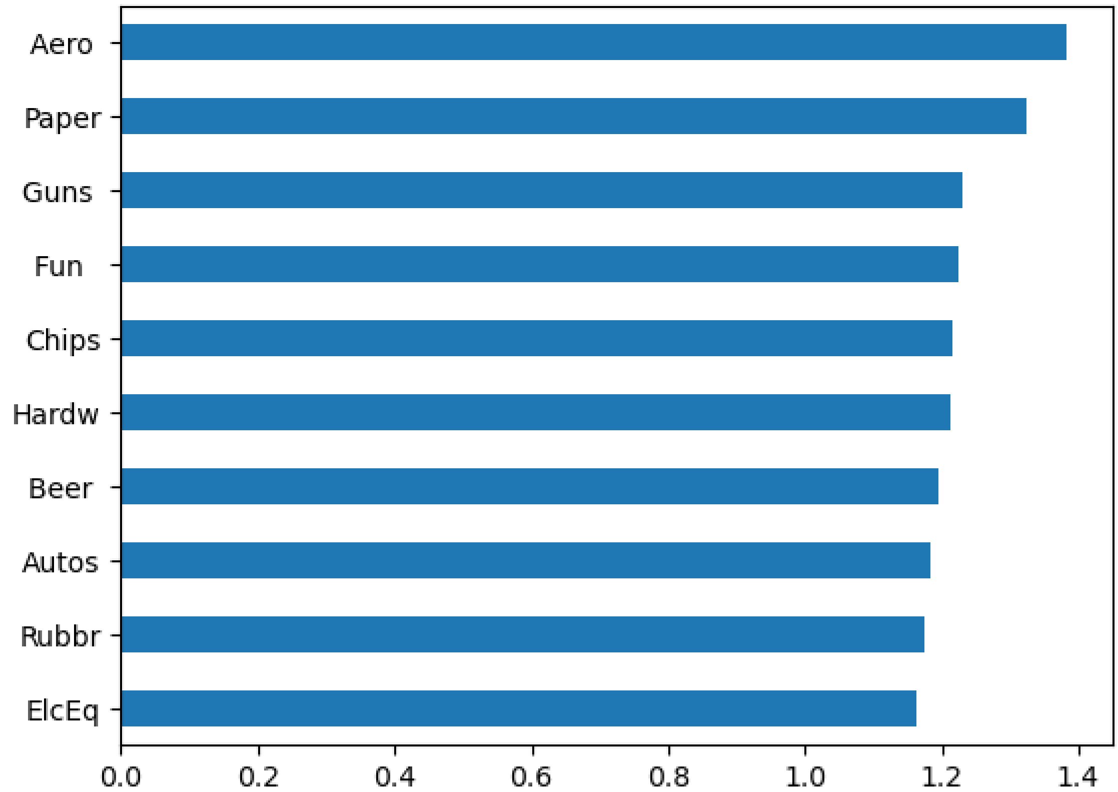
Out[16]:

```
Other    0.727437
```

```
FabPr    0.824516
RlEst    0.832068
Whlsl    0.842667
Telcm    0.843649
Util     0.887124
PerSv    0.912913
Trans    0.923162
Hshld    0.928323
Clths    0.938375
Steel    0.947637
Txtls    0.959652
Food     0.970565
Gold     0.976316
Agric    0.977194
Books    0.985343
Toys     0.988471
BusSv    0.994057
BldMt    1.003545
Ships    1.004353
Hlth     1.010283
Softw    1.021270
Mines    1.034205
Rtail    1.043675
Cnstr    1.043840
Oil      1.045048
Chems    1.054466
Boxes    1.064761
Meals    1.068593
Insur    1.075699
Mach     1.079201
Fin      1.089470
Coal     1.091390
Drugs    1.103110
MedEq    1.134683
Soda     1.142685
Smoke    1.145447
Banks    1.152632
LabEq    1.161633
ElcEq    1.162580
Rubbr    1.173846
Autos    1.183840
Beer     1.195864
Hardw    1.211955
Chips    1.216994
Fun      1.224196
Guns     1.230628
Paper    1.323356
Aero     1.382276
dtype: float64
```

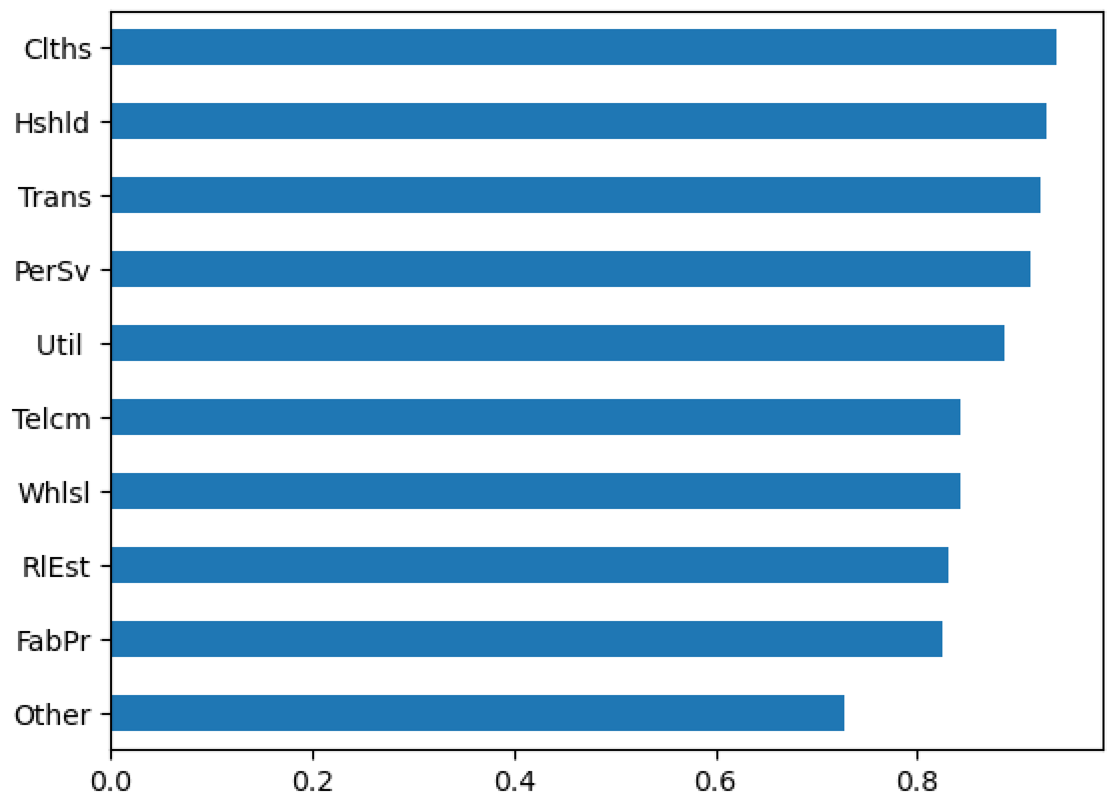
In [17]:
```python
#Plot top 10 histogram
ri_bar.tail(10).plot(kind='barh')
```

Out[17]:
```
<AxesSubplot:>
```



In [18]:
```
#Plot bottom 10 histogram
ri_bar.head(10).plot(kind='barh')
```

Out[18]: <AxesSubplot:>



In [19]:
```
summary_return=value_wted_returns.describe().T
summary_return
```

Out[19]:

	count	mean	std	min	25%	50%	75%	max
Agric	1151.0	0.977194	7.460563	-36.45	-3.065	0.760000	4.760	91.34

Food	1151.0	0.970565	4.753352	-27.87	-1.320	1.070000	3.405	32.63
Soda	1151.0	1.142685	4.861558	-26.26	0.050	1.124394	2.540	38.27
Beer	1151.0	1.195864	7.112490	-29.19	-2.100	1.000000	4.405	87.61
Smoke	1151.0	1.145447	5.814748	-24.93	-2.290	1.300000	4.585	33.04
Toys	1151.0	0.988471	9.902224	-43.34	-4.345	0.830000	5.825	140.45
Fun	1151.0	1.224196	9.264223	-44.28	-3.175	1.260000	5.930	69.57
Books	1151.0	0.985343	7.843385	-34.81	-2.935	0.700000	4.585	54.75
Hshld	1151.0	0.928323	5.761753	-34.97	-1.865	1.050000	4.110	58.33
Clths	1151.0	0.938375	6.143087	-30.85	-2.215	0.960000	4.055	41.40
Hlth	1151.0	1.010283	5.999815	-39.11	0.360	1.010283	1.890	36.47
MedEq	1151.0	1.134683	6.195813	-25.97	-2.285	1.270000	4.720	30.28
Drugs	1151.0	1.103110	5.677732	-35.47	-1.940	1.100000	4.120	39.50
Chems	1151.0	1.054466	6.304059	-33.30	-2.395	1.070000	4.510	46.60
Rubbr	1151.0	1.173846	7.786581	-32.39	-2.175	1.043937	4.575	98.43
Txtls	1151.0	0.959652	7.786859	-35.96	-3.085	1.050000	5.105	58.93
BldMt	1151.0	1.003545	6.915460	-31.81	-2.435	1.240000	4.550	42.41
Cnstr	1151.0	1.043840	9.348882	-38.04	-3.800	0.820000	5.460	67.40
Steel	1151.0	0.947637	8.536370	-32.91	-3.735	1.140000	5.315	80.84
FabPr	1151.0	0.824516	5.839429	-32.50	-0.790	0.830693	2.475	30.38
Mach	1151.0	1.079201	7.215400	-33.35	-2.685	1.490000	4.805	52.08
ElcEq	1151.0	1.162580	7.598979	-34.53	-2.725	1.060000	5.265	59.58
Autos	1151.0	1.183840	8.285257	-36.42	-2.985	0.960000	5.125	81.88
Aero	1151.0	1.382276	9.232740	-40.40	-3.335	1.280000	5.515	72.37
Ships	1151.0	1.004353	8.044326	-34.42	-3.030	1.030000	4.965	63.37
Guns	1151.0	1.230628	5.086981	-30.08	0.055	1.284961	2.630	32.64
Gold	1151.0	0.976316	8.153909	-33.53	-1.410	0.925449	2.690	80.09
Mines	1151.0	1.034205	7.291411	-34.75	-2.955	0.780000	4.970	46.10
Coal	1151.0	1.091390	11.003785	-40.72	-4.500	0.670000	5.915	125.43
Oil	1151.0	1.045048	6.381901	-34.68	-2.430	0.940000	4.560	39.08
Util	1151.0	0.887124	5.473849	-33.05	-1.675	1.060000	3.620	43.46
Telcm	1151.0	0.843649	4.612309	-21.56	-1.510	0.910000	3.255	28.17
PerSv	1151.0	0.912913	9.081922	-39.29	-3.220	0.621921	4.975	84.67
BusSv	1151.0	0.994057	6.911879	-40.28	-2.205	1.220000	4.235	56.83
Hardw	1151.0	1.211955	7.312472	-34.75	-2.785	1.300000	5.250	54.04
Softw	1151.0	1.021270	8.506101	-35.94	-0.735	1.041669	2.665	73.65
Chips	1151.0	1.216994	8.553929	-42.15	-3.370	1.550000	6.040	62.78
LabEq	1151.0	1.161633	6.722871	-33.22	-2.720	1.300000	5.105	25.42
Paper	1151.0	1.323356	15.042804	-62.08	-3.075	0.904520	4.520	300.00

Boxes	1151.0	1.064761	6.077639	-29.24	-2.305	1.180000	4.585	43.19
Trans	1151.0	0.923162	7.073449	-34.61	-2.760	1.020000	4.400	65.40
Whlsl	1151.0	0.842667	7.272389	-43.85	-2.460	1.140000	4.295	57.64
Rtail	1151.0	1.043675	5.956936	-30.41	-2.095	1.010000	4.290	43.51
Meals	1151.0	1.068593	6.477742	-31.61	-2.295	1.250000	4.610	30.65
Banks	1151.0	1.152632	7.020402	-34.00	-2.090	1.200000	4.595	41.79
Insur	1151.0	1.075699	7.375539	-45.76	-2.300	1.010000	4.480	75.11
RlEst	1151.0	0.832068	9.575857	-52.54	-3.590	0.730000	5.025	66.02
Fin	1151.0	1.089470	7.644688	-39.47	-2.730	1.380000	5.025	66.79
Other	1151.0	0.727437	7.290271	-33.56	-2.970	0.790000	4.645	45.30

In [20]:

```
#Inter-quartile Range
iqr=summary_return['75%']-summary_return['25%']
iqr.sort_values()
```

Out[20]:

```
Hlth     1.530
Soda     2.490
Guns     2.575
FabPr    3.265
Softw    3.400
Gold     4.100
Food     4.725
Telcm    4.765
Util     5.295
Hshld    5.975
Drugs    6.060
Clths    6.270
Rtail    6.385
BusSv    6.440
Beer     6.505
Banks    6.685
Rubbr    6.750
Whlsl    6.755
Insur    6.780
Smoke    6.875
Boxes    6.890
Meals    6.905
Chems    6.905
BldMt    6.985
Oil      6.990
MedEq    7.005
Trans    7.160
Mach     7.490
Books    7.520
Paper    7.595
Other    7.615
Fin      7.755
Agric    7.825
LabEq    7.825
Mines    7.925
ElcEq    7.990
Ships    7.995
Hardw    8.035
Autos    8.110
Txtls    8.190
PerSv    8.195
RlEst    8.615
```

```
Aero       8.850
Steel      9.050
Fun        9.105
Cnstr      9.260
Chips      9.410
Toys      10.170
Coal      10.415
dtype: float64
```

In [199…

```
#Correlation analysis
value_wted_returns.iloc[:,1:].corr()
```

Out[199]:

	Agric	Food	Soda	Beer	Smoke	Toys	Fun	Books	Hshld	Clths	...	
Agric	1.000000	0.574797	0.216367	0.453961	0.420412	0.459495	0.555590	0.536256	0.534848	0.434004	...	0.5
Food	0.574797	1.000000	0.412846	0.686322	0.655354	0.542316	0.668993	0.647818	0.774869	0.626494	...	0.7
Soda	0.216367	0.412846	1.000000	0.343991	0.333403	0.254821	0.327058	0.313627	0.395066	0.421101	...	0.3
Beer	0.453961	0.686322	0.343991	1.000000	0.454063	0.607033	0.617797	0.547501	0.661448	0.509862	...	0.5
Smoke	0.420412	0.655354	0.333403	0.454063	1.000000	0.384018	0.473776	0.441969	0.547059	0.429566	...	0.5
Toys	0.459495	0.542316	0.254821	0.607033	0.384018	1.000000	0.615032	0.581242	0.559627	0.538045	...	0.5
Fun	0.555590	0.668993	0.327058	0.617797	0.473776	0.615032	1.000000	0.649313	0.687210	0.620490	...	0.6
Books	0.536256	0.647818	0.313627	0.547501	0.441969	0.581242	0.649313	1.000000	0.644374	0.637933	...	0.6
Hshld	0.534848	0.774869	0.395066	0.661448	0.547059	0.559627	0.687210	0.644374	1.000000	0.596720	...	0.7
Clths	0.434004	0.626494	0.421101	0.509862	0.429566	0.538045	0.620490	0.637933	0.596720	1.000000	...	0.6
Hlth	0.311403	0.376940	0.427470	0.287621	0.281326	0.325558	0.370281	0.374437	0.338893	0.492179	...	0.3
MedEq	0.483310	0.644559	0.340335	0.557070	0.468249	0.527743	0.624478	0.576266	0.648280	0.555829	...	0.6
Drugs	0.557769	0.729804	0.335449	0.622163	0.542798	0.510534	0.635772	0.570664	0.721105	0.515025	...	0.6
Chems	0.601359	0.706716	0.350943	0.614611	0.498769	0.577666	0.701785	0.656437	0.730359	0.660119	...	0.7
Rubbr	0.543954	0.640757	0.322992	0.587193	0.458486	0.565404	0.713794	0.652791	0.657455	0.639424	...	0.6
Txtls	0.512388	0.647941	0.368965	0.601647	0.442467	0.592024	0.714447	0.694056	0.656363	0.726815	...	0.6
BldMt	0.613849	0.747288	0.388123	0.683832	0.527976	0.672076	0.764784	0.737251	0.768605	0.724231	...	0.7
Cnstr	0.497449	0.594044	0.273748	0.551092	0.429212	0.579698	0.646919	0.655441	0.598475	0.621958	...	0.6
Steel	0.566235	0.610976	0.261761	0.533822	0.447807	0.541462	0.696041	0.655096	0.640878	0.603669	...	0.7
FabPr	0.325719	0.288063	0.374589	0.217098	0.222265	0.323909	0.364488	0.354790	0.287127	0.455999	...	0.3
Mach	0.622234	0.685913	0.331612	0.619602	0.485354	0.621873	0.757407	0.721212	0.733254	0.680373	...	0.7
ElcEq	0.614002	0.714241	0.318703	0.612709	0.496086	0.596970	0.746607	0.688423	0.755871	0.635555	...	0.7
Autos	0.506149	0.619449	0.309640	0.542330	0.433205	0.558241	0.696402	0.681622	0.679934	0.649807	...	0.7
Aero	0.475548	0.616188	0.276926	0.563731	0.414795	0.533782	0.639799	0.571091	0.640783	0.576795	...	0.6
Ships	0.529525	0.633231	0.314516	0.544830	0.463441	0.529376	0.630938	0.645323	0.613479	0.589426	...	0.6
Guns	0.261626	0.334804	0.376740	0.237136	0.261440	0.321699	0.314244	0.295100	0.312955	0.424984	...	0.3
Gold	0.112317	0.100500	0.072158	0.070565	0.106177	0.111712	0.094787	0.094781	0.083741	0.119375	...	0.1
Mines	0.548550	0.527516	0.266543	0.484576	0.405681	0.524987	0.618832	0.548160	0.549279	0.544625	...	0.6
Coal	0.443719	0.466590	0.221038	0.420124	0.349853	0.394057	0.515128	0.508152	0.445192	0.448222	...	0.5
Oil	0.514581	0.569146	0.215153	0.484355	0.416206	0.438407	0.551047	0.541989	0.540012	0.475797	...	0.5

Util	0.466754	0.701691	0.238727	0.577151	0.502493	0.451631	0.594581	0.563985	0.639760	0.464850	...	0.5
Telcm	0.439418	0.637033	0.363550	0.507954	0.461676	0.491604	0.622700	0.565786	0.595820	0.534858	...	0.6
PerSv	0.413795	0.539806	0.251959	0.491585	0.378445	0.536659	0.574575	0.524635	0.544717	0.562932	...	0.5
BusSv	0.461705	0.567940	0.339542	0.492871	0.395872	0.517974	0.569147	0.603481	0.564189	0.582024	...	0.5
Hardw	0.485539	0.582723	0.277396	0.512892	0.417691	0.511891	0.664718	0.584905	0.671900	0.573867	...	0.6
Softw	0.298369	0.266009	0.302769	0.230878	0.198115	0.305704	0.386750	0.320591	0.294947	0.455608	...	0.3
Chips	0.531229	0.582269	0.276039	0.558633	0.402276	0.579142	0.705500	0.615986	0.642725	0.604936	...	0.6
LabEq	0.542851	0.587456	0.368578	0.506314	0.440240	0.552202	0.672481	0.560893	0.630328	0.602017	...	0.6
Paper	0.339226	0.445443	0.145153	0.549906	0.310774	0.522802	0.524649	0.437698	0.464765	0.418200	...	0.4
Boxes	0.552172	0.701029	0.363503	0.576661	0.500939	0.532622	0.657333	0.645901	0.712487	0.645266	...	1.0
Trans	0.603829	0.687611	0.318683	0.608854	0.480009	0.621066	0.722917	0.708477	0.683291	0.663292	...	0.7
Whlsl	0.547372	0.676131	0.318357	0.664149	0.482531	0.630704	0.734288	0.664899	0.686742	0.654877	...	0.6
Rtail	0.559494	0.779244	0.377232	0.646874	0.513657	0.603888	0.744741	0.687990	0.763569	0.739077	...	0.7
Meals	0.533303	0.705775	0.416439	0.635849	0.484112	0.639390	0.727745	0.646935	0.692310	0.680269	...	0.6
Banks	0.527050	0.697784	0.359733	0.598558	0.508700	0.548438	0.675223	0.629945	0.677266	0.607806	...	0.6
Insur	0.568157	0.733186	0.314577	0.596377	0.513990	0.488532	0.701952	0.600113	0.695197	0.571042	...	0.6
RlEst	0.490012	0.595841	0.304714	0.548577	0.407512	0.547971	0.663460	0.641775	0.586625	0.618484	...	0.5
Fin	0.592978	0.700761	0.314659	0.617949	0.478121	0.578668	0.763490	0.691074	0.702283	0.644126	...	0.7
Other	0.521455	0.647417	0.328630	0.572498	0.490388	0.578241	0.679032	0.612758	0.671501	0.622855	...	0.6

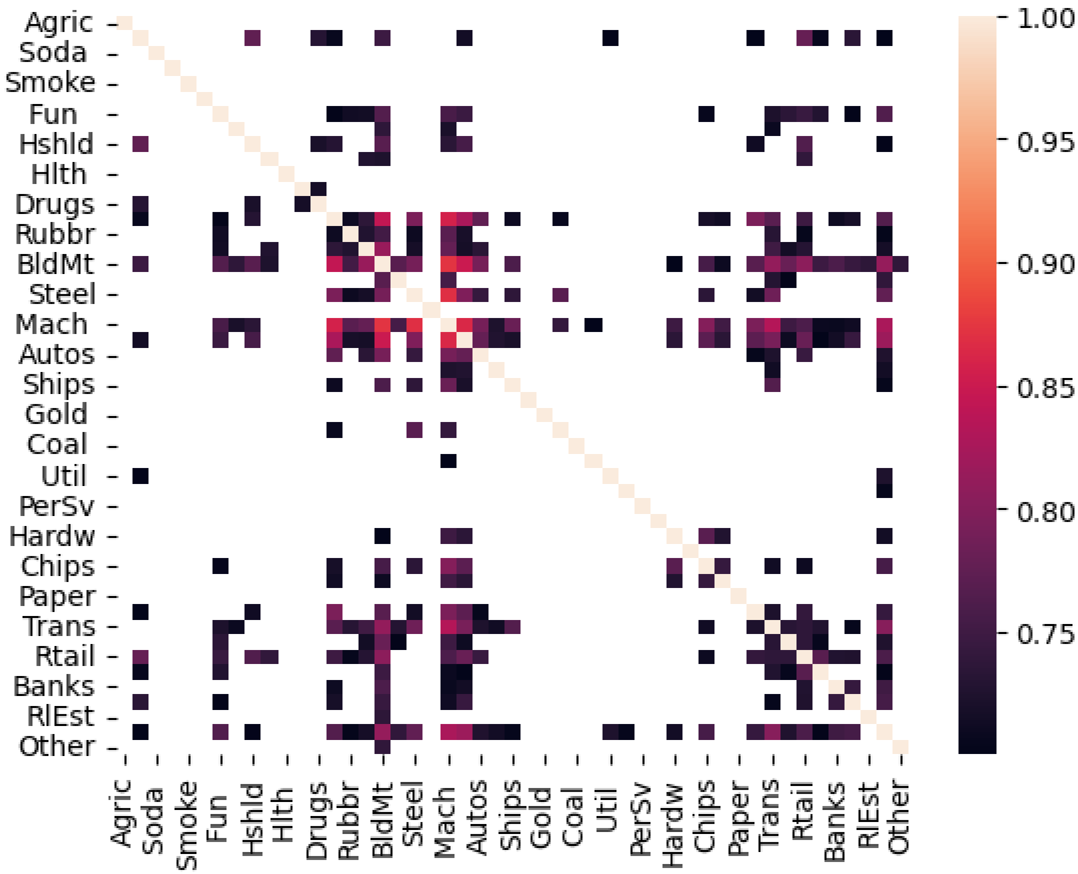
49 rows × 49 columns

In [194…

```
#Only cases with absolute correlation greater than or equal to 0.7
sns.heatmap(value_wted_returns.iloc[:,1:].corr()[(value_wted_returns.iloc[:,1:].corr()>=
```

Out[194]:

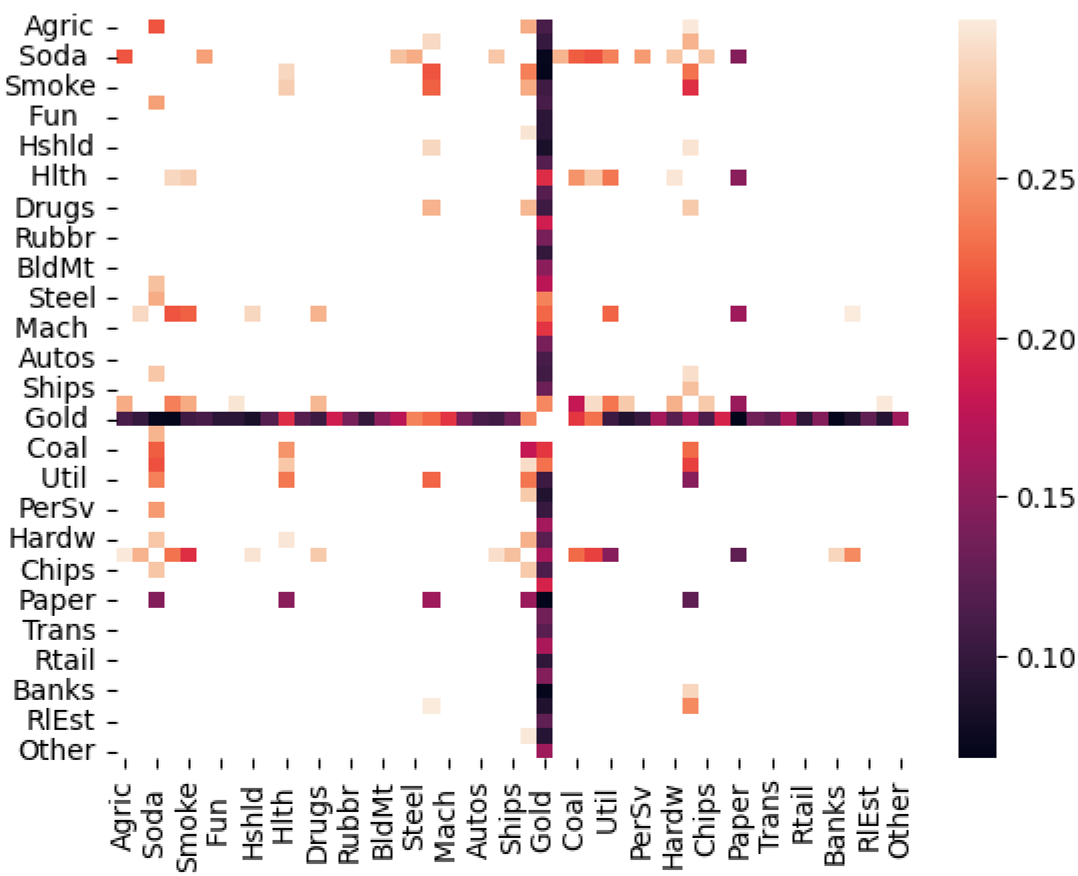
```
<AxesSubplot:>
```



In [38]:

```
#Only cases with absolute correlation less than or equals to 0.3
sns.heatmap(value_wted_returns.iloc[:,1:].corr()[(value_wted_returns.iloc[:,1:].corr()>=
```

Out[38]: <AxesSubplot:>



In [39]:

```
#Conversion to time-series data
l=[]
```

```python
for i in value_wted_returns.columns[1:]:
    df_temp=pd.DataFrame()
    df_temp['Time']=value_wted_returns['Time']
    df_temp['Industry']=i
    df_temp['Industry_Returns']=value_wted_returns[i]
    l.append(df_temp)
df_ts=pd.concat(l,axis=0)
df_ts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 56399 entries, 0 to 1150
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Time              56399 non-null  object 
 1   Industry          56399 non-null  object 
 2   Industry_Returns  56399 non-null  float64
dtypes: float64(1), object(2)
memory usage: 1.7+ MB
```

In [40]:
```python
#Our data gets further reduced in size due to missing data for certain dates in either o
df_fin=pd.merge(df_ts,ff_monthly_data,how='inner',on='Time')
df_fin.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 34643 entries, 0 to 34642
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Time              34643 non-null  object 
 1   Industry          34643 non-null  object 
 2   Industry_Returns  34643 non-null  float64
 3   Mkt-RF            34643 non-null  object 
 4   SMB               34643 non-null  object 
 5   HML               34643 non-null  object 
 6   RMW               34643 non-null  object 
 7   CMA               34643 non-null  object 
 8   RF                34643 non-null  object 
dtypes: float64(1), object(8)
memory usage: 2.6+ MB
```

In [41]:
```python
#Excess return calculations

df_fin[df_fin.columns[3:]]=df_fin[df_fin.columns[3:]].astype(float)
df_fin['re_bar']=df_fin['Industry_Returns']-df_fin['RF']
df_fin.head(10)
```

Out[41]:

	Time	Industry	Industry_Returns	Mkt-RF	SMB	HML	RMW	CMA	RF	re_bar
0	196307	Agric	3.04	-0.39	-0.44	-0.89	0.68	-1.23	0.27	2.77
1	196307	Food	-0.46	-0.39	-0.44	-0.89	0.68	-1.23	0.27	-0.73
2	196307	Soda	2.57	-0.39	-0.44	-0.89	0.68	-1.23	0.27	2.30
3	196307	Beer	-2.19	-0.39	-0.44	-0.89	0.68	-1.23	0.27	-2.46
4	196307	Smoke	-2.54	-0.39	-0.44	-0.89	0.68	-1.23	0.27	-2.81
5	196307	Toys	-5.07	-0.39	-0.44	-0.89	0.68	-1.23	0.27	-5.34
6	196307	Fun	-0.70	-0.39	-0.44	-0.89	0.68	-1.23	0.27	-0.97
7	196307	Books	-0.07	-0.39	-0.44	-0.89	0.68	-1.23	0.27	-0.34
8	196307	Hshld	-0.15	-0.39	-0.44	-0.89	0.68	-1.23	0.27	-0.42
9	196307	Clths	-0.67	-0.39	-0.44	-0.89	0.68	-1.23	0.27	-0.94

In [44]:
```
#Industry wise calculation
risk_return_df=df_fin.groupby('Industry').agg({'Industry_Returns':['mean','std']})
risk_return_df.head(10)
```

Out[44]:

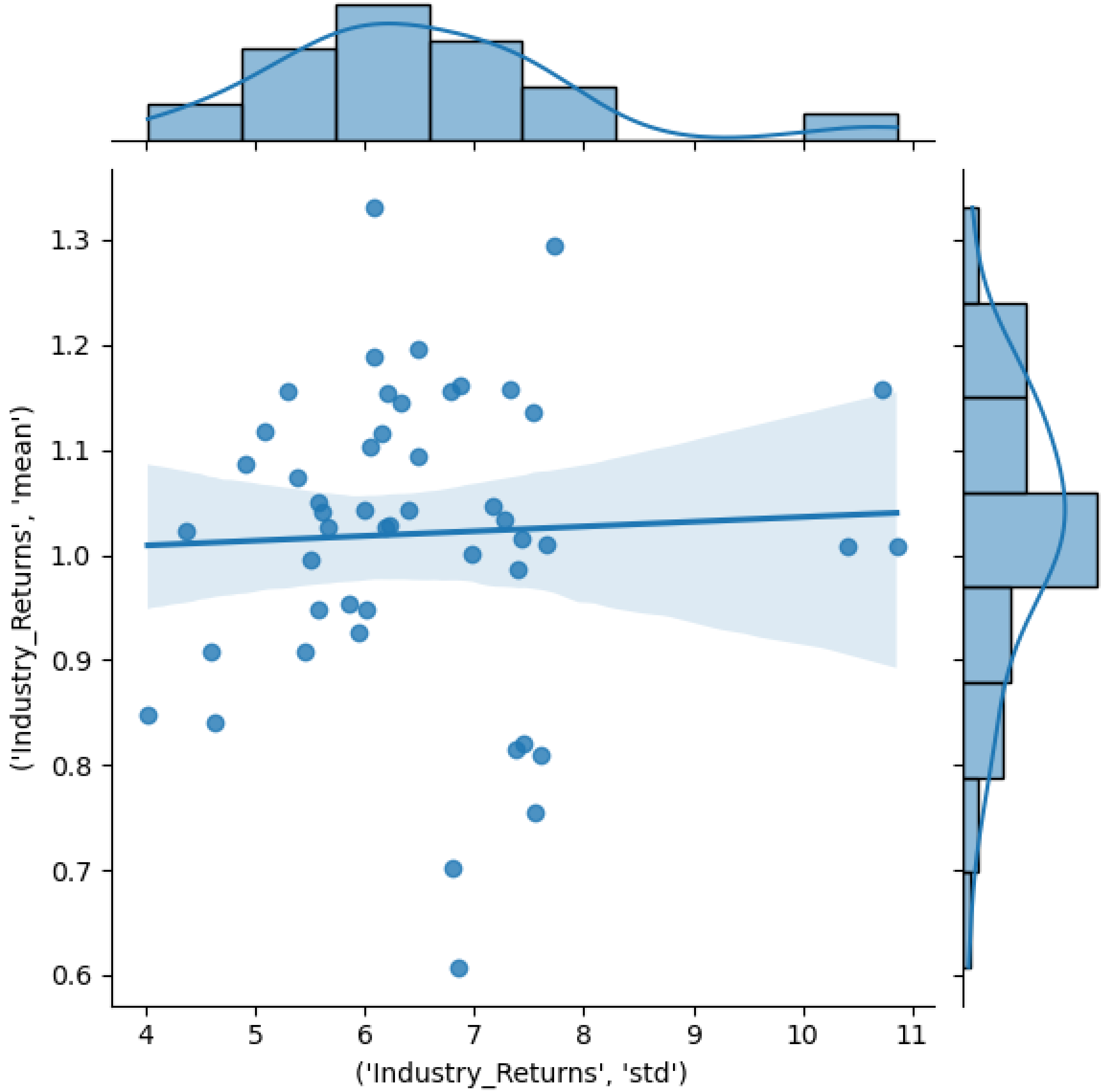
	Industry_Returns	
	mean	std
Industry		
Aero	1.160594	6.873825
Agric	1.042801	6.401401
Autos	1.015502	7.435402
Banks	0.947836	6.012328
Beer	1.117949	5.086936
BldMt	1.026040	6.191058
Books	0.925502	5.939802
Boxes	0.995615	5.511701
BusSv	1.026294	5.669321
Chems	0.948741	5.580444

In [45]:
```
risk_return_df.shape
```

Out[45]:
```
(49, 2)
```

In [46]:
```
sns.jointplot(y=risk_return_df[('Industry_Returns', 'mean')],x=risk_return_df[('Industry
```

Out[46]:
```
<seaborn.axisgrid.JointGrid at 0x29a897ea3a0>
```



Traditional Risk-Return paradigm states that risk and the corresponding return go hand in hand, that is, higher the risk corresponds to higher reward. We can clearly observe quite a lot of deviation from the ideal case. This deviation can be attributed to the long time horizon and the different natures of all the different industries and their corresponding market circumstances. All in all, we observe a positive slope which is in agreement with the ideal case dispite having such a diverse data sample.

1. FF 3 factor model's industry wise application

a) Industry wise

In [47]:
```
l=[]
for i in df_fin['Industry'].unique():
    df_temp=df_fin[df_fin['Industry']==i]
    X=df_fin[df_fin['Industry']==i][['Mkt-RF','HML','SMB']]
    y=df_fin[df_fin['Industry']==i]['re_bar']
    model=LinearRegression()
    model.fit(X,y)
    l.append([i,model.intercept_,model.coef_[0],model.coef_[1],model.coef_[2]])
```

In [48]:
```
output_parameters=pd.DataFrame(l,columns=['Industry','alpha','beta_mke-rf','beta_hml','b
output_parameters.set_index('Industry',inplace=True)
output_parameters.head(10)
```

Out[48]:

Industry	alpha	beta_mke-rf	beta_hml	beta_smb
Agric	0.123052	0.792252	0.054621	0.419211
Food	0.224902	0.728281	0.190194	-0.158193
Soda	0.274125	0.866519	0.205056	-0.161487
Beer	0.319567	0.777741	0.081535	-0.130624
Smoke	0.532657	0.731028	0.249511	-0.249968
Toys	-0.295823	1.093581	0.035539	0.543507
Fun	0.106524	1.273701	0.003684	0.473940
Books	-0.178282	1.063937	0.243308	0.293418
Hshld	0.123085	0.818153	-0.017683	-0.151367
Clths	-0.027471	1.069713	0.221113	0.388554

We test for which all industries do we have a significant α value.

Statistical test for positive alpha values:

$$H_o : \alpha <= 0$$

$$H_a : \alpha > 0$$

We do a t-test for above hypothesis. It will be a right tailed test at 95% confidence interval.

In [50]:

```
df_t_test=output_parameters['alpha'].to_frame()
df_t_test['t_score']=(df_t_test['alpha']-df_t_test['alpha'].mean())/(df_t_test['alpha'].
df_t_test
```

Out[50]:

Industry	alpha	t_score
Agric	0.123052	0.576462
Food	0.224902	0.997931
Soda	0.274125	1.201621
Beer	0.319567	1.389671
Smoke	0.532657	2.271469
Toys	-0.295823	-1.156911
Fun	0.106524	0.508066
Books	-0.178282	-0.670509
Hshld	0.123085	0.576595
Clths	-0.027471	-0.046428
Hlth	-0.007516	0.036148
MedEq	0.377056	1.627569
Drugs	0.409976	1.763794
Chems	-0.135012	-0.491449

Rubbr	0.004920	0.087609
Txtls	-0.350674	-1.383892
BldMt	-0.198565	-0.754443
Cnstr	-0.203660	-0.775524
Steel	-0.483452	-1.933346
FabPr	-0.315638	-1.238907
Mach	-0.093855	-0.321133
ElcEq	0.068233	0.349609
Autos	-0.233029	-0.897060
Aero	-0.009518	0.027863
Ships	-0.141694	-0.519102
Guns	0.208567	0.930332
Gold	0.292011	1.275638
Mines	-0.028855	-0.052154
Coal	-0.034739	-0.076504
Oil	0.017448	0.139456
Util	0.082224	0.407506
Telcm	0.001827	0.074811
PerSv	-0.374036	-1.480567
BusSv	0.015098	0.129729
Hardw	0.140494	0.648637
Softw	-0.035897	-0.081298
Chips	0.165183	0.750804
LabEq	0.206508	0.921815
Paper	-0.127222	-0.459211
Boxes	0.064744	0.335170
Trans	-0.154208	-0.570883
Whlsl	-0.006968	0.038419
Rtail	0.160553	0.731646
Meals	0.172796	0.782308
Banks	-0.269482	-1.047906
Insur	-0.001626	0.060521
RlEst	-0.623108	-2.511268
Fin	-0.050892	-0.143348
Other	-0.506653	-2.029359

In [51]:

```
t_stat_critical_value=s.t.ppf(q=0.95,df=48)
t_stat_critical_value
```

Out[51]: 1.6772241953450393

In [200…
```
df_t_test['Reject_Ho']=np.where((df_t_test['t_score']>t_stat_critical_value)|(df_t_test[
df_t_test[df_t_test['Reject_Ho']==1]
```

Out[200]:

Industry	alpha	t_score	Reject_Ho
Smoke	0.532657	2.271469	1
Drugs	0.409976	1.763794	1
Steel	-0.483452	-1.933346	1
RlEst	-0.623108	-2.511268	1
Other	-0.506653	-2.029359	1

b) Overall market

In [53]:
```
# R regression approach for overall market

df_2nd_pass=df_fin.groupby('Industry').mean()['re_bar']
df_2nd_pass=df_2nd_pass.sort_index()

df_final=output_parameters.sort_index()
df_final['re_bar']=df_2nd_pass
df_final.drop(columns=['alpha'],inplace=True)
df_final
```

Out[53]:

Industry	beta_mke-rf	beta_hml	beta_smb	re_bar
Aero	1.154123	0.326350	0.246603	0.796973
Agric	0.792252	0.054621	0.419211	0.679180
Autos	1.265263	0.454733	0.137214	0.651881
Banks	1.194491	0.665083	-0.119846	0.584215
Beer	0.777741	0.081535	-0.130624	0.754328
BldMt	1.199291	0.401560	0.272023	0.662419
Books	1.063937	0.243308	0.293418	0.561881
Boxes	0.977340	0.098625	-0.064160	0.631994
BusSv	1.064510	-0.129574	0.398365	0.662673
Chems	1.106349	0.334070	-0.033097	0.585120
Chips	1.244688	-0.471083	0.333086	0.793508
Clths	1.069713	0.221113	0.388554	0.730071
Cnstr	1.235851	0.216194	0.498143	0.670198
Coal	1.104526	0.340065	0.453170	0.793748
Drugs	0.811561	-0.265418	-0.281450	0.722405
ElcEq	1.211425	0.022051	0.106481	0.781273
FabPr	1.021359	0.167546	0.655578	0.457016

Fin	1.249810	0.247246	0.102387	0.752999
Food	0.728281	0.190194	-0.158193	0.659533
Fun	1.273701	0.003684	0.473940	0.930184
Gold	0.508354	-0.043782	0.360953	0.644639
Guns	0.836448	0.350894	0.197723	0.832885
Hardw	1.098486	-0.490863	0.140310	0.637652
Hlth	0.965435	-0.025345	0.533935	0.646663
Hshld	0.818153	-0.017683	-0.151367	0.545134
Insur	1.039844	0.419258	-0.130485	0.686025
LabEq	1.113354	-0.450969	0.450837	0.793126
Mach	1.174607	0.086558	0.311329	0.663819
Meals	1.012593	0.077212	0.262841	0.825545
MedEq	0.857152	-0.281932	0.086301	0.791344
Mines	1.106386	0.318476	0.357801	0.772871
Oil	0.944245	0.491733	-0.104552	0.679533
Other	1.138192	0.154611	0.276869	0.244074
Paper	1.014149	0.332004	-0.012956	0.544767
PerSv	1.035001	0.061052	0.499222	0.338571
RlEst	1.154797	0.547170	0.872141	0.390835
Rtail	0.977786	-0.046554	0.061144	0.710552
Rubbr	0.991946	0.145007	0.588126	0.739066
Ships	1.139721	0.442853	0.204747	0.683706
Smoke	0.731028	0.249511	-0.249968	0.967016
Soda	0.866519	0.205056	-0.161487	0.790552
Softw	1.282164	-0.753334	0.872137	0.644837
Steel	1.283837	0.369072	0.409657	0.445488
Telcm	0.837358	0.146211	-0.191829	0.476789
Toys	1.093581	0.035539	0.543507	0.451641
Trans	1.083384	0.284701	0.205318	0.590325
Txtls	1.118299	0.633929	0.661026	0.623041
Util	0.606420	0.336512	-0.200872	0.484314
Whlsl	0.988903	0.060981	0.485809	0.676676

In [54]:

```
X=sm.add_constant(df_fin[['Mkt-RF', 'HML','SMB']])
y=df_fin['re_bar']

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

```
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:                 re_bar   R-squared:                       0.504
Model:                            OLS   Adj. R-squared:                  0.504
Method:                 Least Squares   F-statistic:                 1.172e+04
Date:                Tue, 07 Feb 2023   Prob (F-statistic):               0.00
Time:                        21:45:54   Log-Likelihood:            -1.0285e+05
No. Observations:               34643   AIC:                         2.057e+05
Df Residuals:                   34639   BIC:                         2.057e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0163      0.026     -0.631      0.528      -0.067       0.034
Mkt-RF         1.0278      0.006    169.123      0.000       1.016       1.040
HML            0.1396      0.009     15.861      0.000       0.122       0.157
SMB            0.2279      0.009     26.130      0.000       0.211       0.245
==============================================================================
Omnibus:                     7824.968   Durbin-Watson:                   1.576
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           142739.118
Skew:                           0.612   Prob(JB):                         0.00
Kurtosis:                      12.869   Cond. No.                         4.78
==============================================================================

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

Based on above, we can say that when the whole market in considered at once, then we find that all three coefficients are significant and Market risk premium dominates the other two factors in dependent variable's sensitivity to it.

2. Correlation between Automobile stocks

In [235…
```python
import yfinance as yf
from pandas.plotting import scatter_matrix
from datetime import datetime
```

In [222…
```python
datetime.now().year
```

Out[222]:
```
2023
```

In [228…
```python
symbol = 'TSLA'
ticker = yf.Ticker(symbol)
tesla = ticker.history(period='1y',
interval='1d',
actions=True,
auto_adjust=True)
tesla.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 252 entries, 2022-02-07 00:00:00-05:00 to 2023-02-07 00:00:00-05:00
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Open          252 non-null    float64
 1   High          252 non-null    float64
 2   Low           252 non-null    float64
 3   Close         252 non-null    float64
 4   Volume        252 non-null    int64  
 5   Dividends     252 non-null    float64
 6   Stock Splits  252 non-null    float64
dtypes: float64(6), int64(1)
memory usage: 15.8 KB
```

In [229…

```
symbol = 'F'
ticker = yf.Ticker(symbol)
ford = ticker.history(period='1y',
interval='1d',
actions=True,
auto_adjust=True)
ford.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 252 entries, 2022-02-07 00:00:00-05:00 to 2023-02-07 00:00:00-05:00
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Open          252 non-null    float64
 1   High          252 non-null    float64
 2   Low           252 non-null    float64
 3   Close         252 non-null    float64
 4   Volume        252 non-null    int64
 5   Dividends     252 non-null    float64
 6   Stock Splits  252 non-null    float64
dtypes: float64(6), int64(1)
memory usage: 15.8 KB
```

In [230…

```
symbol = 'GM'
ticker = yf.Ticker(symbol)
gm = ticker.history(period='1y',
interval='1d',
actions=True,
auto_adjust=True)
gm.info()
```

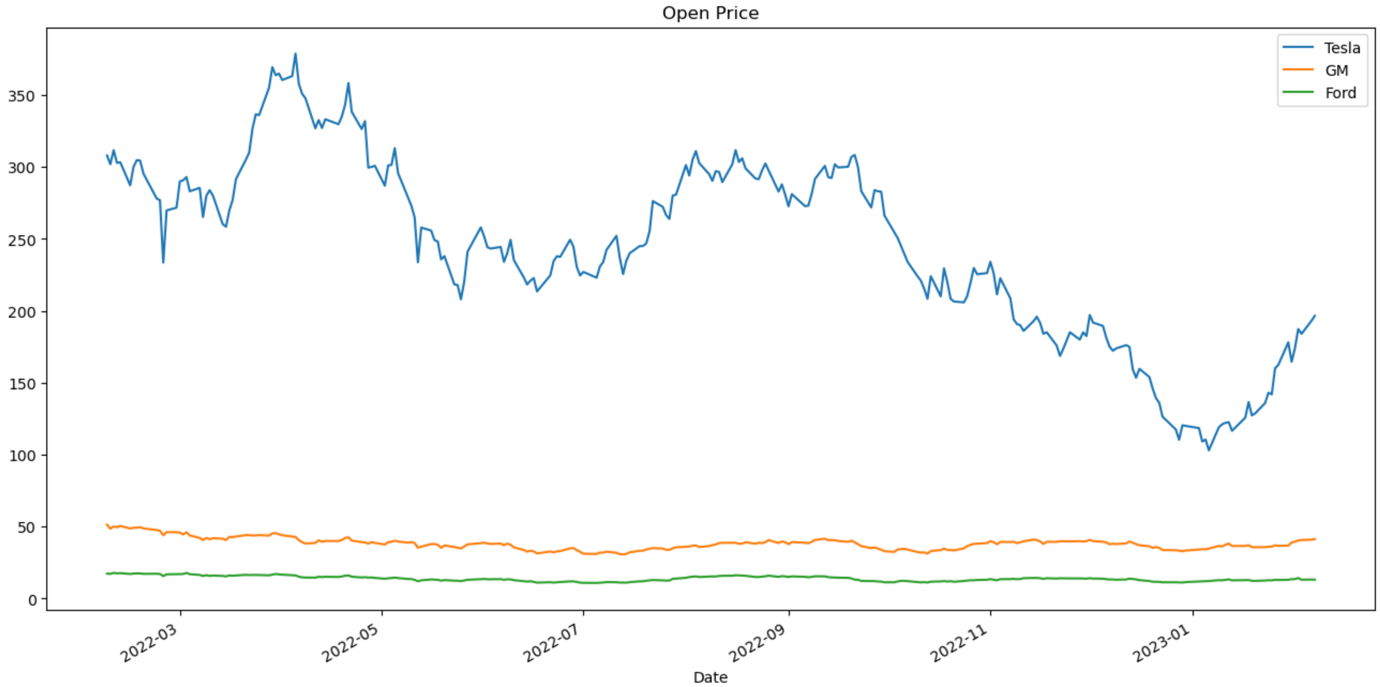
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 252 entries, 2022-02-07 00:00:00-05:00 to 2023-02-07 00:00:00-05:00
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Open          252 non-null    float64
 1   High          252 non-null    float64
 2   Low           252 non-null    float64
 3   Close         252 non-null    float64
 4   Volume        252 non-null    int64
 5   Dividends     252 non-null    float64
 6   Stock Splits  252 non-null    float64
dtypes: float64(6), int64(1)
memory usage: 15.8 KB
```

In [231…

```
tesla['Open'].plot(label='Tesla',figsize=(16,8),title='Open Price')
gm['Open'].plot(label='GM')
ford['Open'].plot(label='Ford')
plt.legend()
```

Out[231]:

```
<matplotlib.legend.Legend at 0x29a8f6bde50>
```

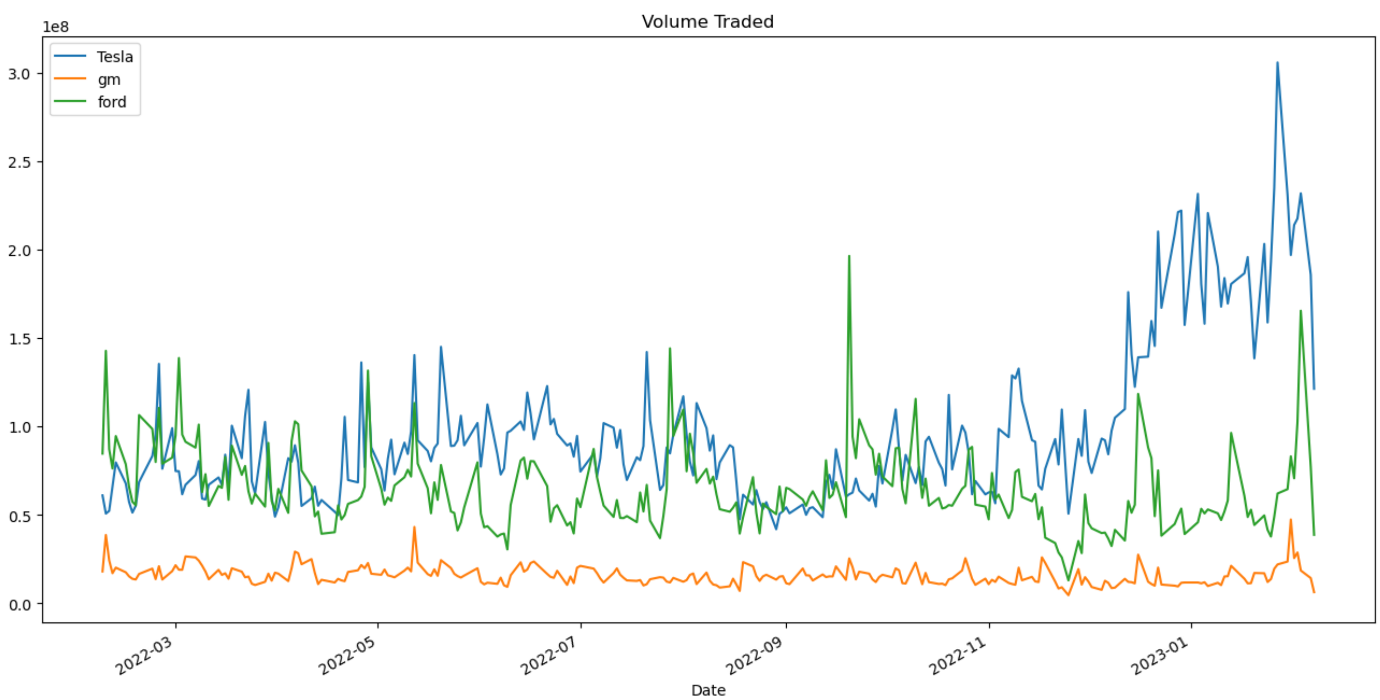


In [232…

```
tesla['Volume'].plot(label='Tesla',figsize=(16,8),title='Volume Traded')
gm['Volume'].plot(label='gm')
ford['Volume'].plot(label='ford')
plt.legend()
```

Out[232]:

```
<matplotlib.legend.Legend at 0x29a91164e80>
```



In [233…

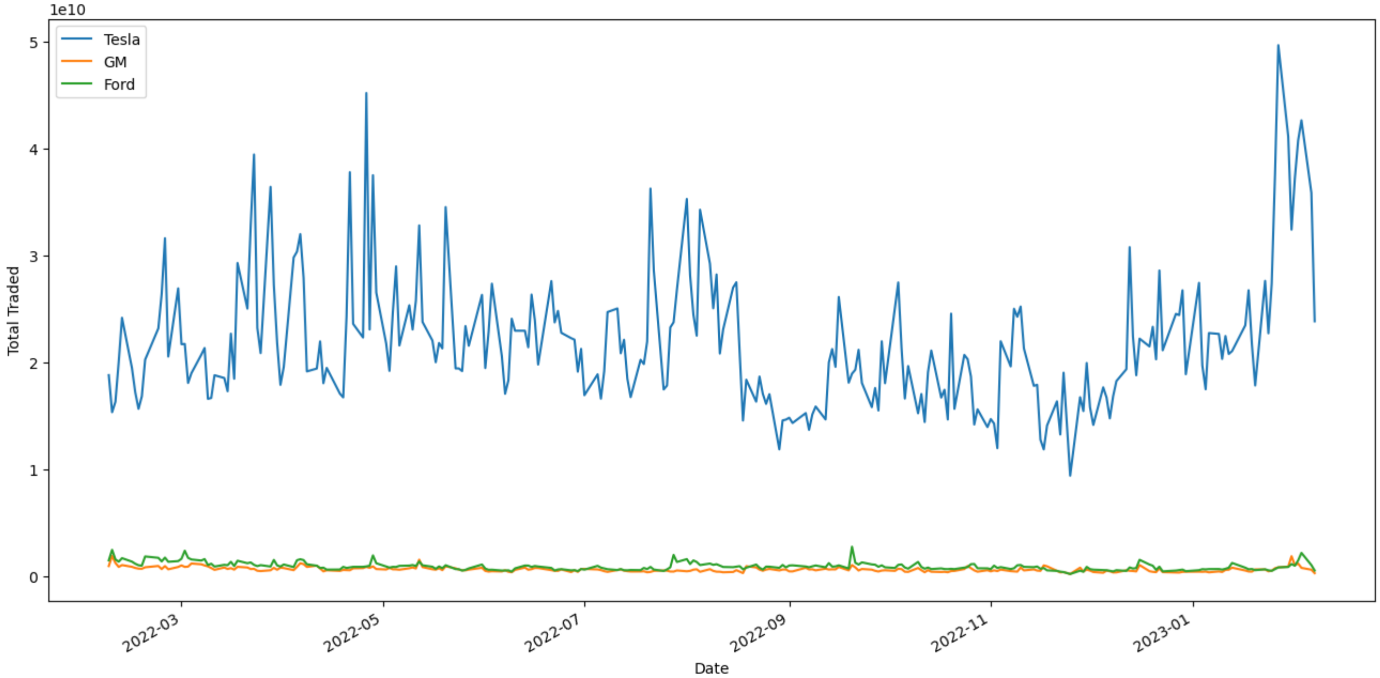
```
tesla['Total Traded'] = tesla['Open']*tesla['Volume']
ford['Total Traded'] = ford['Open']*ford['Volume']
gm['Total Traded'] = gm['Open']*gm['Volume']
```

In [241…

```
tesla['Total Traded'].plot(label='Tesla',figsize=(16,8))
gm['Total Traded'].plot(label='GM')
ford['Total Traded'].plot(label='Ford')
plt.legend()
plt.ylabel('Total Traded')
```

Out[241]:

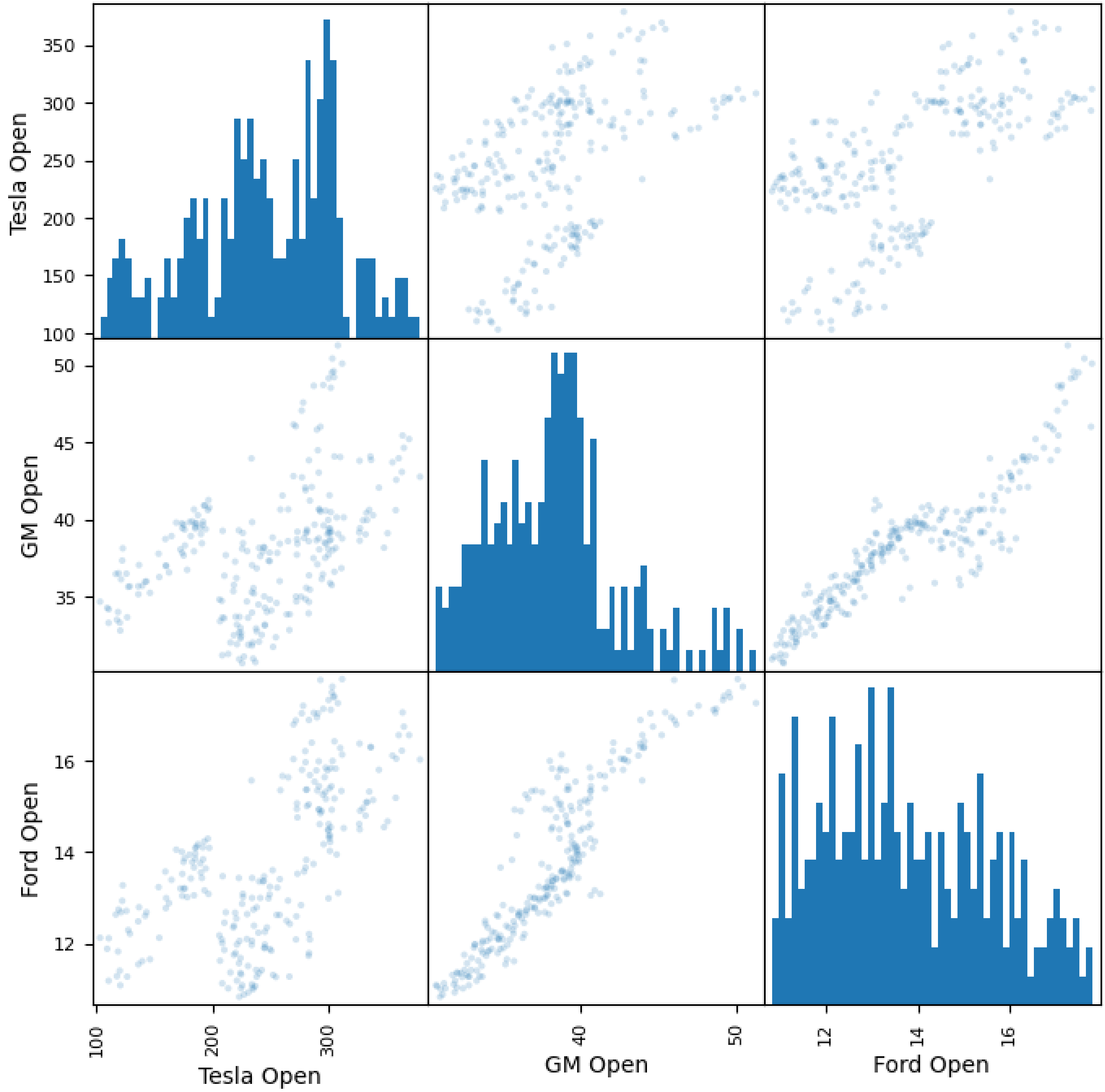
```
Text(0, 0.5, 'Total Traded')
```



In [236…
```
comparison = pd.concat([tesla['Open'],gm['Open'],ford['Open']],axis=1)
comparison.columns = ['Tesla Open','GM Open','Ford Open']

scatter_matrix(comparison,figsize=(8,8),alpha=0.2,hist_kwds={'bins':50})
```

Out[236]:
```
array([[<AxesSubplot:xlabel='Tesla Open', ylabel='Tesla Open'>,
        <AxesSubplot:xlabel='GM Open', ylabel='Tesla Open'>,
        <AxesSubplot:xlabel='Ford Open', ylabel='Tesla Open'>],
       [<AxesSubplot:xlabel='Tesla Open', ylabel='GM Open'>,
        <AxesSubplot:xlabel='GM Open', ylabel='GM Open'>,
        <AxesSubplot:xlabel='Ford Open', ylabel='GM Open'>],
       [<AxesSubplot:xlabel='Tesla Open', ylabel='Ford Open'>,
        <AxesSubplot:xlabel='GM Open', ylabel='Ford Open'>,
        <AxesSubplot:xlabel='Ford Open', ylabel='Ford Open'>]],
      dtype=object)
```



In [251…:
```
comparison.describe().loc['mean',:]/comparison.describe().loc['std',:]
```

Out[251]:
```
Tesla Open     3.900512
GM Open        9.155948
Ford Open      7.624141
dtype: float64
```

In [237…:
```
tesla['Open'].corr(ford['Open'])
```

Out[237]:
```
0.6315345206188344
```

In [238…:
```
tesla['Open'].corr(gm['Open'])
```

Out[238]:
```
0.4539597606104765
```

In [239…:
```
gm['Open'].corr(ford['Open'])
```

Out[239]:
```
0.9065758928982761
```

In [255…:
```
comparison_2 = pd.concat([tesla['Close'],gm['Close'],ford['Close']],axis=1)
```

```python
comparison_2.columns = ['Tesla Close','GM Close','Ford Close']

ret=comparison_2.pct_change().dropna()
ret.describe()
```

Out[255]:

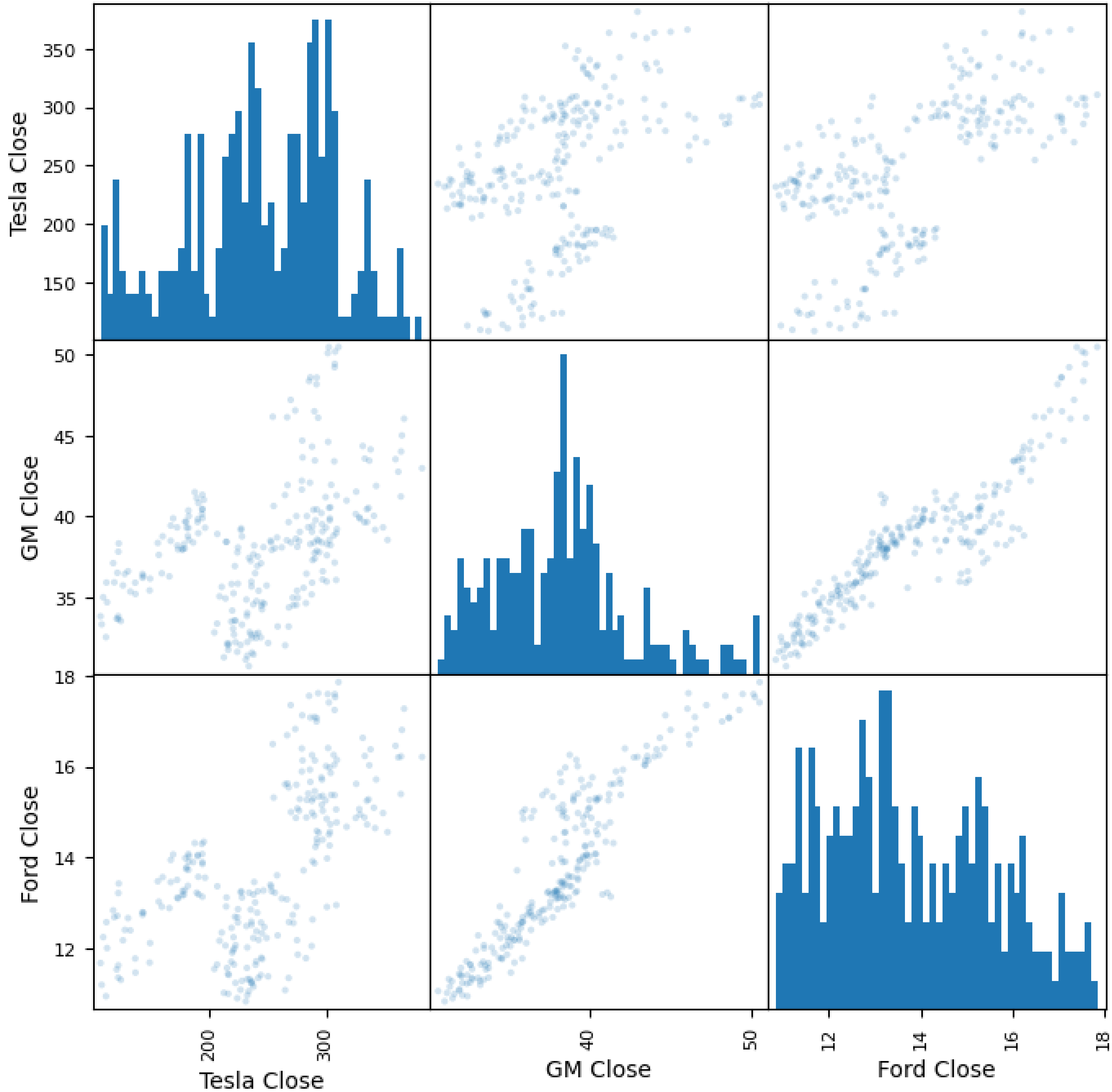
	Tesla Close	GM Close	Ford Close
count	251.000000	251.000000	251.000000
mean	-0.000914	-0.000459	-0.000671
std	0.042503	0.027837	0.029467
min	-0.122422	-0.080749	-0.123242
25%	-0.025651	-0.019388	-0.020458
50%	0.000974	0.000985	0.001460
75%	0.023823	0.017643	0.018458
max	0.110002	0.089139	0.085209

In [256…

```python
scatter_matrix(comparison_2,figsize=(8,8),alpha=0.2,hist_kwds={'bins':50})
```

Out[256]:

```
array([[<AxesSubplot:xlabel='Tesla Close', ylabel='Tesla Close'>,
        <AxesSubplot:xlabel='GM Close', ylabel='Tesla Close'>,
        <AxesSubplot:xlabel='Ford Close', ylabel='Tesla Close'>],
       [<AxesSubplot:xlabel='Tesla Close', ylabel='GM Close'>,
        <AxesSubplot:xlabel='GM Close', ylabel='GM Close'>,
        <AxesSubplot:xlabel='Ford Close', ylabel='GM Close'>],
       [<AxesSubplot:xlabel='Tesla Close', ylabel='Ford Close'>,
        <AxesSubplot:xlabel='GM Close', ylabel='Ford Close'>,
        <AxesSubplot:xlabel='Ford Close', ylabel='Ford Close'>]],
      dtype=object)
```


Tesla Close
GM Close
Ford Close
350
300
250
200
150
50
45
40
35
18
16
14
12
200
300
40
50
12
14
16
18
Tesla Close
GM Close
Ford Close

In [ ]: