AP HW4

Efficient Frontier Revisited

> ### Part 1: Minimum-Tracking-Error Frontier

In [32]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
import statsmodels.api as sm
```

In [33]:
```
industry_portfolios = \
    pd\
    .read_csv("/Users/lu/Desktop/Industry_Portfolios.csv")

risk_factor =\
    pd\
    .read_csv("/Users/lu/Desktop/Risk_Factors.csv")

data = pd.merge(industry_portfolios, risk_factor, on='Date')
```

In [34]:
```
# excess return of market portfolio
industry_columns = ['NoDur', 'Durbl', 'Manuf', 'Enrgy', 'HiTec', 'Telcm',
for industry in industry_columns:
    data[industry] = data[industry] - data['Rf']
```

expected deviation:

$$R_i = E(\tilde{R}_i - \tilde{R}_m)$$

In [35]:
```
# expected deviation from market return
expected_deviations = {}
for industry in industry_columns:
    expected_deviations [industry] = (data[industry] - data['Rm-Rf']).mea

df = pd.DataFrame(expected_deviations.items(), columns=['Industry', 'Expe
df
```

Out[35]:

	Industry	Expected Deviation
0	NoDur	0.154750
1	Durbl	-0.014750
2	Manuf	0.264750
3	Enrgy	0.483083
4	HiTec	0.018167
5	Telcm	0.133333
6	Shops	0.168250
7	Hlth	0.035750
8	Utils	0.159083
9	Other	-0.259000

In [36]:

```
num_points = len(df['Expected Deviation'])
R_p = np.linspace(0, 0.1, num_points)
```

covariance matrix of return deviations:

$$V_{ij} = \mathrm{Cov}[(\tilde{R}_i - \tilde{R}_m), (\tilde{R}_j - \tilde{R}_m)]$$

In [37]:

```
deviation_matrix = data[industry_columns].subtract(data['Rm-Rf'], axis=0)

cov_matrix = deviation_matrix.cov()
cov_matrix_df = pd.DataFrame(cov_matrix, columns=industry_columns, index=

cov_matrix_df
```

Out[37]:

	NoDur	Durbl	Manuf	Enrgy	HiTec	Telcm	Sho
NoDur	5.439696	-6.073035	-1.396192	-1.200533	-1.883151	1.538885	1.140
Durbl	-6.073035	26.628901	4.908024	-3.481055	1.891577	-1.707625	-0.3543
Manuf	-1.396192	4.908024	2.950499	1.666133	0.065267	-0.626416	-1.1545
Enrgy	-1.200533	-3.481055	1.666133	19.274911	-1.516972	-1.040525	-3.7104
HiTec	-1.883151	1.891577	0.065267	-1.516972	5.098746	-0.773294	-0.2453
Telcm	1.538885	-1.707625	-0.626416	-1.040525	-0.773294	4.682567	0.4637
Shops	1.140741	-0.354335	-1.154597	-3.710439	-0.245350	0.463797	4.4526
Hlth	3.815137	-8.082946	-2.288900	-2.485796	-1.936284	0.693157	0.7645
Utils	4.272002	-9.617490	-1.901412	4.454368	-2.342839	2.721477	-0.1766
Other	-1.768738	4.385865	0.358904	-3.864826	-1.404050	-1.271778	-0.2569

In [38]:
```
R = np.array(list(expected_deviations.values()), dtype=float)
V = cov_matrix_df.values
e = np.ones(len(expected_deviations))
# print(type(R))
```

In [39]:
```
inv_V = np.linalg.inv(V)
```

In [57]:
```
alpha = np.dot(np.dot(R.T, inv_V), e)
print("\nalpha=",alpha)
zeta = np.dot(np.dot(R.T, inv_V), R)
print("\nzeta =",zeta)
delta = np.dot(np.dot(e.T, inv_V), e)
print("\ndelta =",delta)
```

```
alpha= 2.9321278826306285

zeta = 0.2047449735113006

delta = 58.550254376399124
```

Weights for Minimum-Tracking-Error Portfolio:

$$w* = \frac{V^{-1}R}{\mathbf{1}^T V^{-1} R}$$

In [41]:
```
weights = np.dot(inv_V, R) / np.dot(np.dot(e, inv_V), R)
weights_df = pd.DataFrame(weights, index=industry_columns, columns=['Weig
weights_df
```

Out[41]:

	Weights
NoDur	0.052634
Durbl	0.000153
Manuf	0.137627
Enrgy	0.087032
HiTec	0.179353
Telcm	0.071074
Shops	0.106884
Hlth	0.102776
Utils	0.040162
Other	0.222304

Tracking Error:

$$\text{Tracking Error} = \sqrt{\text{variance}}$$

In [42]:
```
variance = np.diag(cov_matrix)
df['Tracking Error'] = np.sqrt(variance)
df
```

Out [42]:

	Industry	Expected Deviation	Tracking Error
0	NoDur	0.154750	2.332316
1	Durbl	-0.014750	5.160320
2	Manuf	0.264750	1.717702
3	Enrgy	0.483083	4.390320
4	HiTec	0.018167	2.258040
5	Telcm	0.133333	2.163924
6	Shops	0.168250	2.110125
7	Hlth	0.035750	2.796506
8	Utils	0.159083	3.502496
9	Other	-0.259000	2.122075

Plot the minimum-tracking-error frontier generated by the ten industry portfolios.

In [43]:

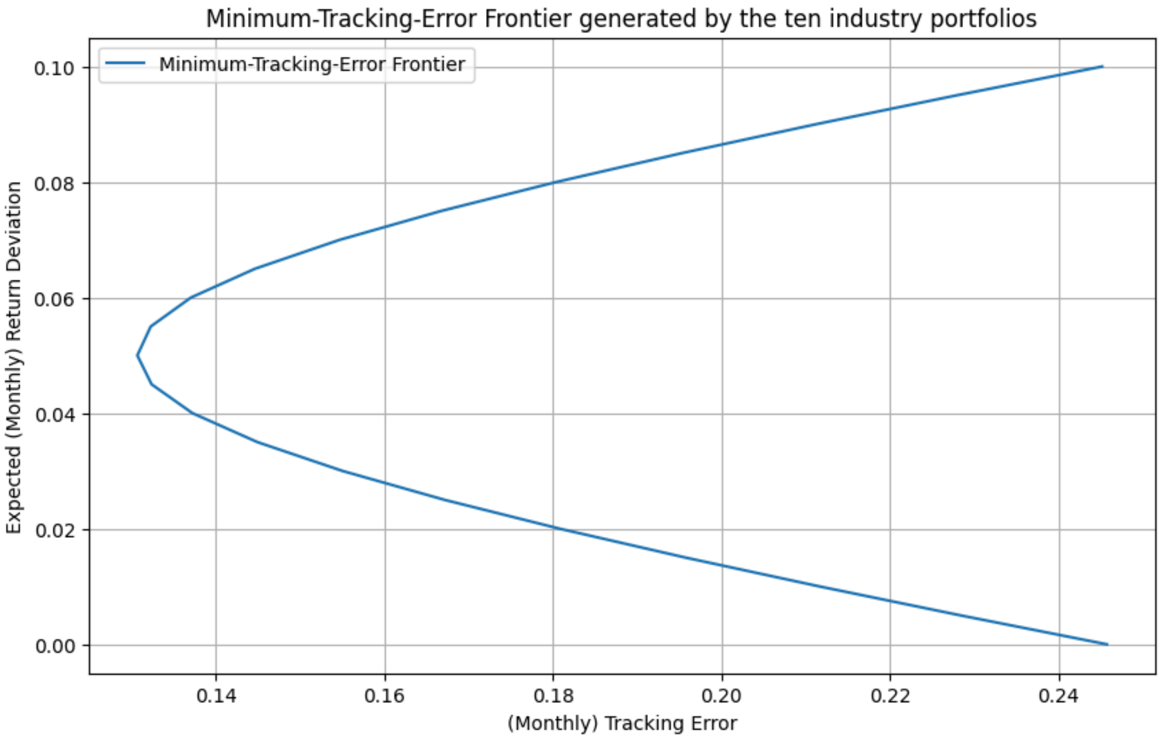
```
R_p = np.arange(0, 0.101, 0.005)

min_track_error_frontier = pd.DataFrame(R_p, columns=['Rp'])

min_track_error_frontier['sd'] = np.sqrt(1 / delta + (delta / (zeta * del
# min_track_error_frontier
```

In [44]:

```
plt.figure(figsize=(10, 6))
plt.plot(min_track_error_frontier['sd'], min_track_error_frontier['Rp'],
plt.xlabel('(Monthly) Tracking Error')
plt.ylabel('Expected (Monthly) Return Deviation')
plt.title('Minimum-Tracking-Error Frontier generated by the ten industry
plt.grid(True)
plt.legend()
plt.show()
```



Information Ratio:

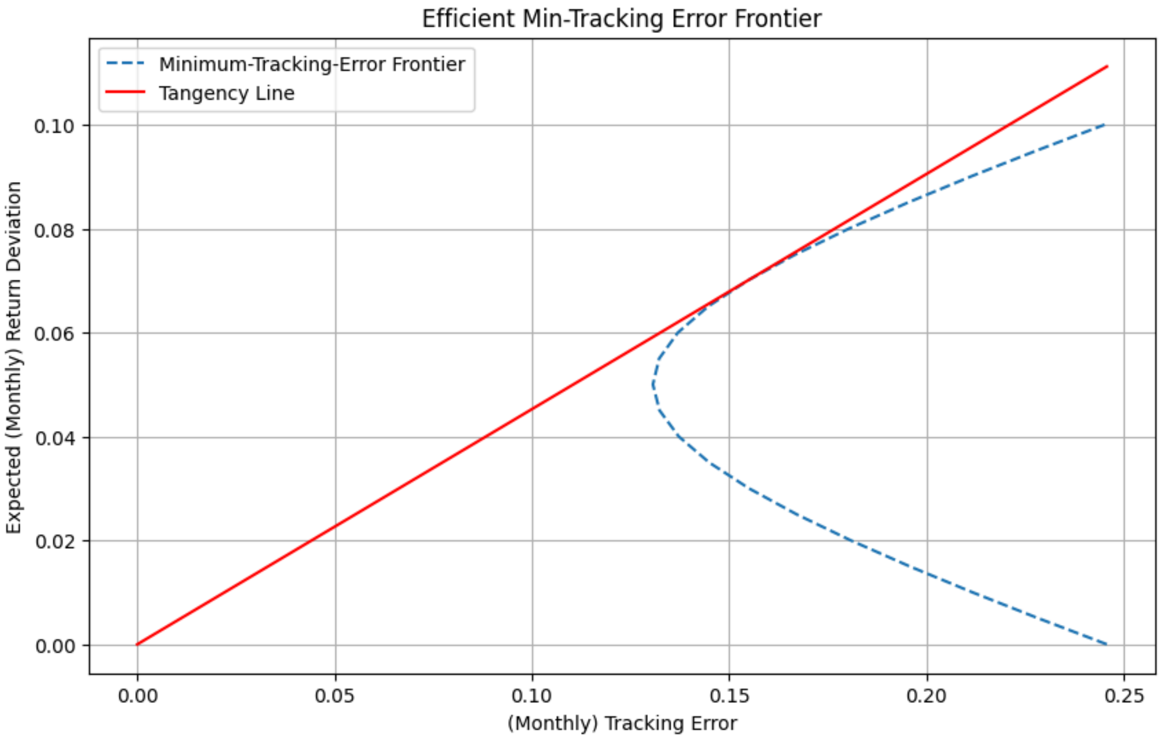
$$\text{IR} = \frac{\text{Expected Deviation}}{\text{Tracking Error}}$$

In [45]:
```
min_track_error_frontier['IR'] = min_track_error_frontier['Rp'] / min_tra
# find the max of IR that is tangency portfolio
tangency_portfolio = min_track_error_frontier.loc[min_track_error_frontie
information_ratio = tangency_portfolio['IR']
print("\ninformation_ratio =", information_ratio)
```

```
information_ratio = 0.45248408073659774
```

In [72]:
```
plt.figure(figsize=(10, 6))
plt.plot(min_track_error_frontier['sd'], min_track_error_frontier['Rp'],

# Tangency Line
slope = tangency_portfolio['IR']# IR
x_vals = np.linspace(0, max(min_track_error_frontier['sd']), 100)
y_vals = slope * x_vals
plt.plot(x_vals, y_vals, label='Tangency Line', color='red')
plt.xlabel('(Monthly) Tracking Error')
plt.ylabel('Expected (Monthly) Return Deviation')
plt.title('Efficient Min-Tracking Error Frontier')
plt.grid(True)
plt.legend()
plt.show()
```



Part 2: Minimum-Variance Frontier w/o Short Sales

In [65]:
```
new_data =\
    industry_portfolios.drop(columns = ['Date'])
mean_returns = new_data[industry_columns].mean()
mean_returns_df = pd.DataFrame(mean_returns, columns=['Mean Return'])
# mean_returns_df
```

In [66]:
```
cov_matrix = data[industry_columns].cov()
cov_matrix_df = pd.DataFrame(cov_matrix, columns=industry_columns, index=
# cov_matrix_df
```

In [67]:
```
def generate_weights(num_portfolios, num_assets):
    weights = np.random.rand(num_portfolios, num_assets)
    weights /= weights.sum(axis=1)[:, np.newaxis]
    return weights
```

In [54]:
```
def portfolio_statistics(weights, mean_returns_df, cov_matrix_df):
    mean_return = np.dot(weights, mean_returns_df)
    portfolio_variance = np.dot(weights.T, np.dot(cov_matrix_df, weights)
    portfolio_std_dev = np.sqrt(portfolio_variance)
    return mean_return, portfolio_std_dev
```

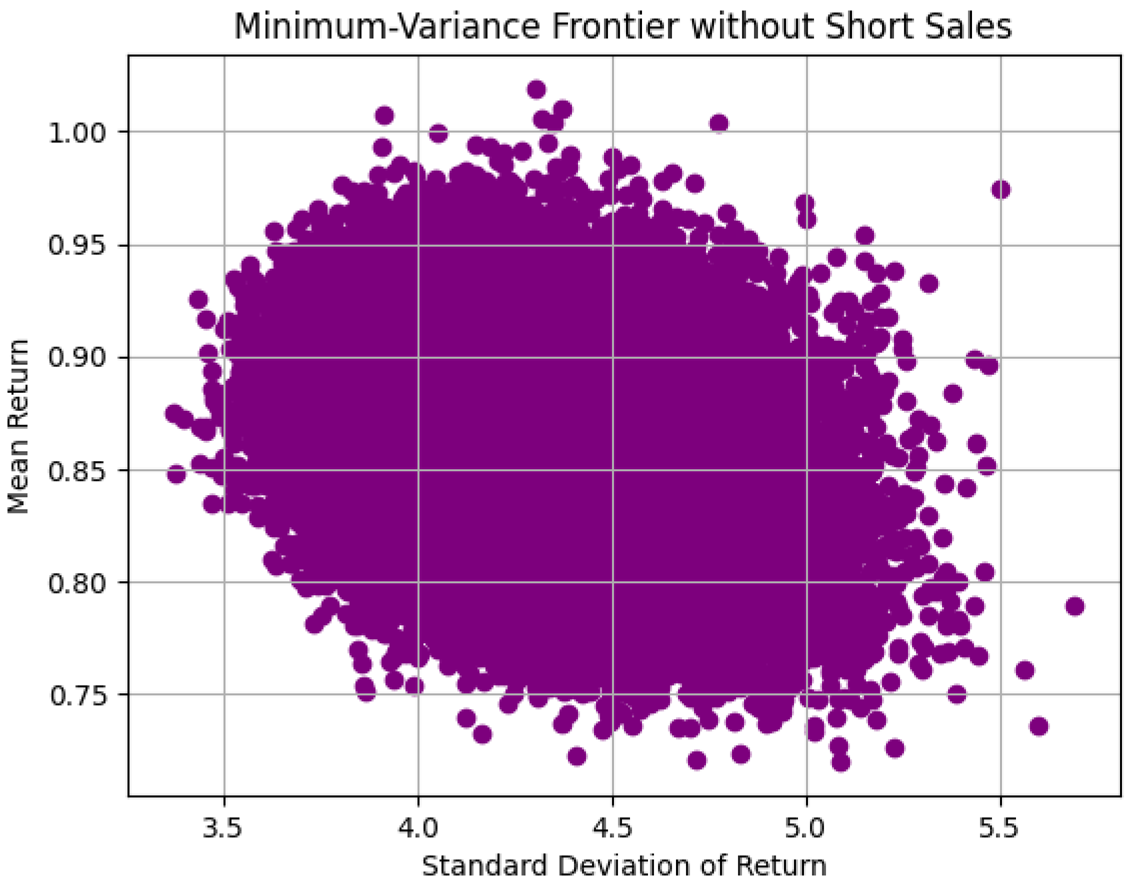
In [55]:
```
num_portfolios = int(1e5)
num_assets = 10

weights = generate_weights(num_portfolios, num_assets)
portfolio_returns = []
portfolio_risks = []

# mean and standard deviation for each portfolio
for w in weights:
    mean_return, std_dev = portfolio_statistics(w, mean_returns_df, cov_m
    portfolio_returns.append(mean_return)
```

```
    portfolio_risks.append(std_dev)

plt.scatter(portfolio_risks, portfolio_returns, c='purple', marker='o')
plt.xlabel('Standard Deviation of Return')
plt.ylabel('Mean Return')
plt.title('Minimum-Variance Frontier without Short Sales')
plt.grid(True)
plt.show()
```

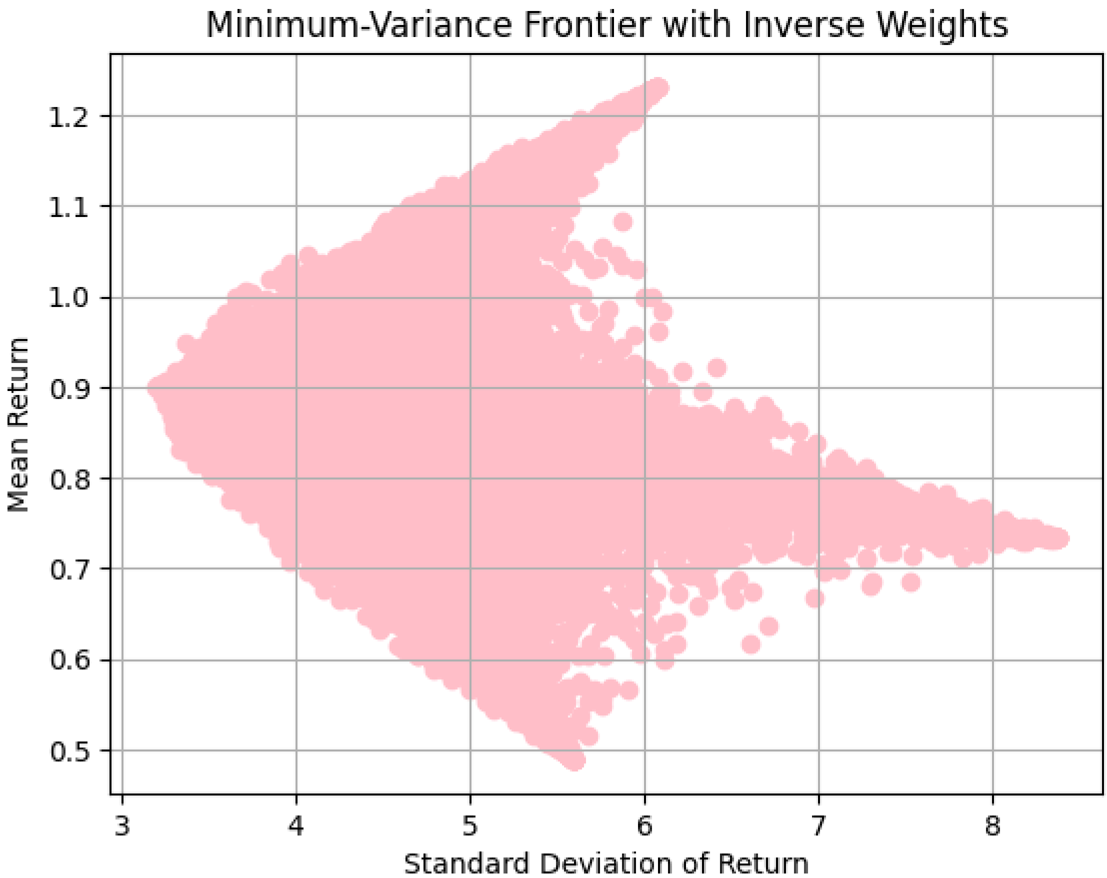


In [56]:
```
inverse_weights = 1 / np.random.rand(num_portfolios, num_assets)
inverse_weights /= inverse_weights.sum(axis=1)[:, np.newaxis]

portfolio_returns_inv = []
portfolio_risks_inv = []

for w in inverse_weights:
    mean_return, std_dev = portfolio_statistics(w, mean_returns_df, cov_m
    portfolio_returns_inv.append(mean_return)
    portfolio_risks_inv.append(std_dev)

plt.scatter(portfolio_risks_inv, portfolio_returns_inv, c='pink', marker=
plt.xlabel('Standard Deviation of Return')
plt.ylabel('Mean Return')
plt.title('Minimum-Variance Frontier with Inverse Weights')
plt.grid(True)
plt.show()
```


Minimum-Variance Frontier with Inverse Weights
Mean Return
Standard Deviation of Return
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
3
4
5
6
7
8