QF600 ASSETING PRCING

HomeWork 1 EFFICIENT FRONTIER

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

In [4]:
```
%whos
```

```
Variable                Type        Data/Info
--------------------------------------------
dataframe_columns       function    <function dataframe_columns at 0x10fc74
ea0>
dataframe_hash          function    <function dataframe_hash at 0x10fc0e700
>
dtypes_str              function    <function dtypes_str at 0x10fc74b80>
get_dataframes          function    <function get_dataframes at 0x12d5e3560
>
getpass                 module      <module 'getpass' from '/<...>b/python3
.12/getpass.py'>
hashlib                 module      <module 'hashlib' from '/<...>b/python3
.12/hashlib.py'>
import_pandas_safely    function    <function import_pandas_safely at 0x12d
71d440>
is_data_frame           function    <function is_data_frame at 0x12d71d1c0>
json                    module      <module 'json' from '/usr<...>on3.12/js
on/__init__.py'>
np                      module      <module 'numpy' from '/us<...>kages/num
py/__init__.py'>
pd                      module      <module 'pandas' from '/u<...>ages/pand
as/__init__.py'>
plt                     module      <module 'matplotlib.pyplo<...>es/matplo
tlib/pyplot.py'>
```

In [5]:
```
data =\
    pd.read_csv("/Users/lu/Desktop/Industry_Portfolios.csv")
```

Q1.1 Table Creation for the mean return and standard deviation of return.

In [7]:
```
data_1 =\
    data.drop(columns = ['Date'])
```

In [8]:
```
mean_returns =\
    data_1.mean()
```

```
sd_returns =\
    data_1.std()

cov_matrix =\
    data_1.cov()

summary = pd.concat([mean_returns, sd_returns], axis = 1)
summary.columns = ['MEAN RETURN', 'STANDARD DEVIATION RETURN']

summary
```

Out[8]:

	MEAN RETURN	STANDARD DEVIATION RETURN
NoDur	0.902833	3.345657
Durbl	0.733333	8.361852
Manuf	1.012833	5.310270
Enrgy	1.231167	6.081524
HiTec	0.766250	5.381191
Telcm	0.881417	4.448284
Shops	0.916333	4.093786
Hlth	0.783833	3.787172
Utils	0.907167	3.701763
Other	0.489083	5.582452

Q1.2 Minimum-Variance Frontier

In [10]:

```
cov_matrix
```

Out[10]:

	NoDur	Durbl	Manuf	Enrgy	HiTec	Telcm	Sl
NoDur	11.193422	18.449666	14.104907	10.531341	12.922949	11.968078	10.170
Durbl	18.449666	69.920577	39.178097	27.019794	35.466652	27.490543	27.44
Manuf	14.104907	39.178097	28.198970	23.145380	24.618739	19.550150	17.62
Enrgy	10.531341	27.019794	23.145380	36.984933	19.267276	15.366817	11.29
HiTec	12.922949	35.466652	24.618739	19.267276	28.957220	18.708273	17.83
Telcm	11.968078	27.490543	19.550150	15.366817	18.708273	19.787227	14.16
Shops	10.170832	27.444731	17.622867	11.297800	17.837115	14.169356	16.75
Hlth	9.953112	16.824003	13.596447	9.630327	13.254064	11.506599	10.17
Utils	7.866653	12.746136	11.440612	14.027168	10.304187	10.991596	6.69
Other	14.438409	39.361987	26.313423	18.320469	23.855470	19.610836	19.22

In [11]:
```
mu = data.mean().values
cov_matrix_1 = np.linalg.inv(cov_matrix)
```

In [12]:
```
R = np.array(mean_returns)
V = np.array(cov_matrix)
e = np.ones(len(mean_returns))
```

In [13]:
```
inv_V = np.linalg.inv(V)
```

In [14]:
```
alpha = np.dot(np.dot(R.T, inv_V), e)
print(alpha)
zeta = np.dot(np.dot(R.T, inv_V), R)
print(zeta)
delta = np.dot(np.dot(e.T, inv_V), e)
print(delta)
```

```
0.13794323869931885
0.19640858464482272
0.1373875973567116
```

$$\sigma_p^2 = \frac{1}{\delta} + \frac{\delta}{\zeta\delta - \alpha^2}(r_p - r_{mv})^2$$

In [16]:
```
expected_returns = np.arange(0, 2, 0.1)
```

In [17]:
```
portfolio_var = []
for r_p in expected_returns:
    #print("r_p: ", r_p)
    # variance = (delta * r_p**2 - 2 * zeta * r_p + alpha) / (alpha * del
    variance = 1. / delta + (delta / (zeta * delta - alpha**2)) * ( ( r_p
```

```
    portfolio_var.append(variance)
#print("check : ", portfolio_var )
portfolio_sd = np.sqrt(portfolio_var)
```

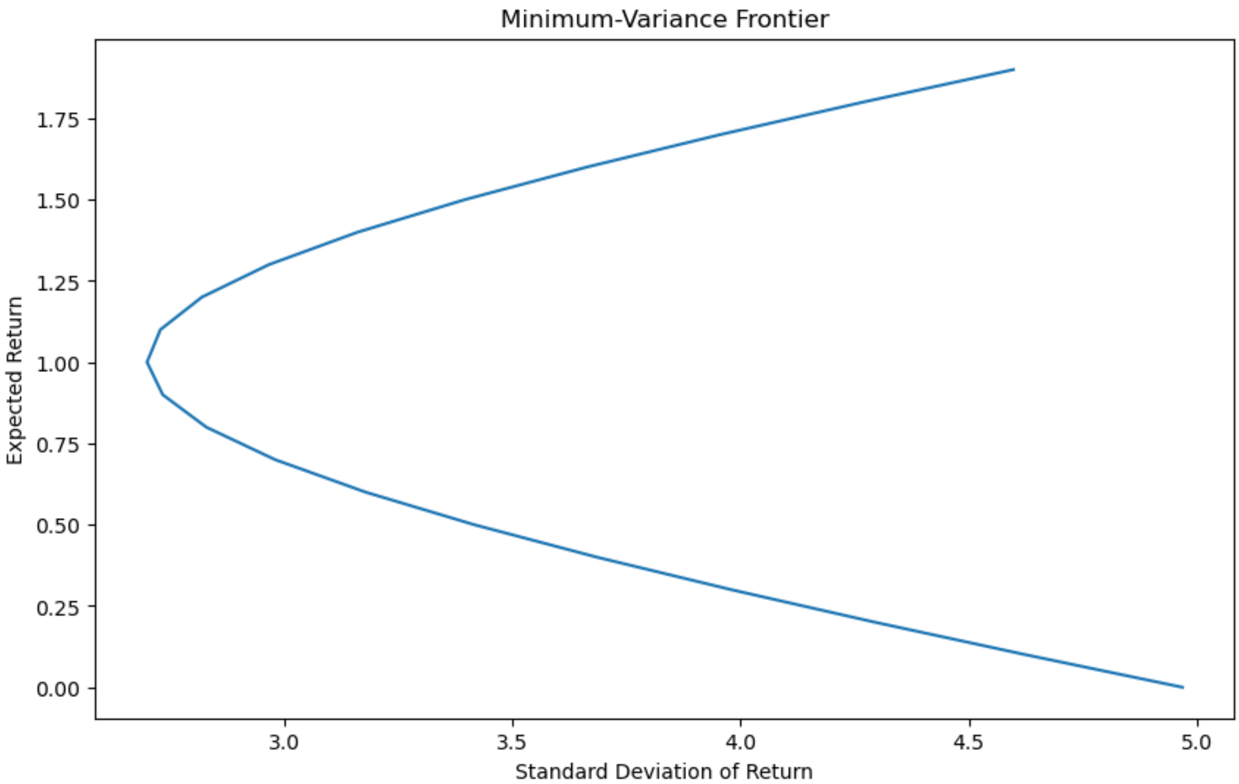
In [18]:
```
print(portfolio_sd)
```

```
[4.96865933 4.62520378 4.29451081 3.97976317 3.68504889 3.41555771
 3.17771336 2.97910534 2.82801233 2.73232821 2.69795476 2.7272112
 2.81811651 2.96500574 3.16008166 3.39504812 3.66223513 3.9551181
 4.26841081 4.59794314]
```

In [19]:
```
print(portfolio_var)
```

```
[24.68757549425759, 21.392509964329186, 18.442823093673947, 15.83851488229
1879, 13.57958533018298, 11.666034437347252, 10.09786220378469, 8.87506862
94953, 7.997653714479077, 7.465617458736025, 7.2789598622661424, 7.4376809
250694285, 7.941780647145885, 8.79125902849551, 9.986116069118305, 11.5263
51769014266, 13.411966128183401, 15.642959146625707, 18.219330824341174, 2
1.141081161329815]
```

In [20]:
```
plt.figure(figsize=(10, 6))
plt.plot(portfolio_sd, expected_returns, label='Minimum-Variance Frontier
plt.xlabel('Standard Deviation of Return')
plt.ylabel('Expected Return')
plt.title('Minimum-Variance Frontier')
plt.show()
```



Q1.3 Explanation

The minimum variance frontier can help the investor build the optimal portfolio, which gets more returns with less risk. The investor can reduce some uncertainty of risk and improve the overall return of their portfolios by diversifying investment.

Q2.1 Support that the risk_free rate is 0.13% per month to Plot The Efficient Frontier

In [23]:
```
r_f = 0.13
```

$$\sigma_p^2 = \frac{(r_p - r_f)^2}{\zeta - 2\alpha r_f + \delta(r_f)^2}$$

In [25]:
```
var = []
for r_p in expected_returns:
    sigma_p_sq = ((r_p - r_f)**2 / (zeta - (2 * alpha * r_f) + delta * (r
    var.append(sigma_p_sq)
print(var)
```

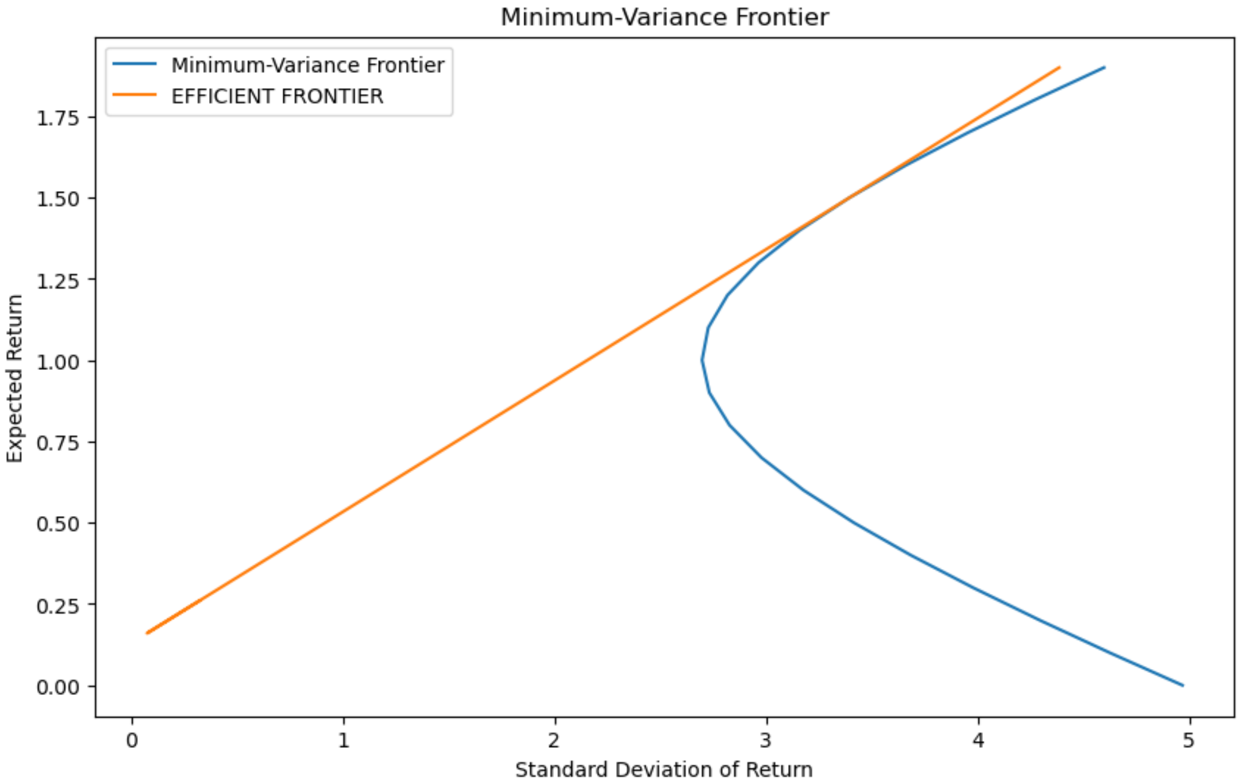
```
[0.32212854665893675, 0.07433735692129309, 0.17345383281635054, 0.42124502
255399426, 0.6690362122916378, 0.9168274020292814, 1.1646185917669252, 1.4
124097815045689, 1.6602009712422126, 1.907992160979856, 2.1557833507174995
, 2.4035745404551436, 2.6513657301927878, 2.89915691993043, 3.146948109668
074, 3.394739299405718, 3.642530489143362, 3.8903216788810058, 4.138112868
618649, 4.385904058356292]
```

$$r_p = r_f + (\zeta - 2\alpha r_f + \delta(r_f^2))^{\frac{1}{2}}\sigma_p$$

In [27]:
```
eff_var = []
for sd_p in var:
    r_p_2 = r_f + (zeta - 2 * alpha * r_f + (delta * (r_f ** 2))) **.5 *
    eff_var.append(r_p_2)
print(eff_var)
```

```
[0.26, 0.16, 0.2, 0.30000000000000004, 0.4, 0.5, 0.6000000000000001, 0.700
0000000000001, 0.8000000000000002, 0.9000000000000001, 1.0, 1.1, 1.2000000
000000006, 1.2999999999999998, 1.4, 1.5, 1.6000000000000005, 1.70000000000
00006, 1.8000000000000003, 1.9000000000000004]
```

In [28]:
```
plt.figure(figsize=(10, 6))
plt.plot(portfolio_sd, expected_returns, label='Minimum-Variance Frontier
plt.plot(var, eff_var, label = 'EFFICIENT FRONTIER')
plt.xlabel('Standard Deviation of Return')
plt.ylabel('Expected Return')
plt.title('Minimum-Variance Frontier')
plt.legend()
plt.show()
```



Q2.2 Explanation

The efficient frontier will provide a Scientific decision to help investors choose the optimal portfolio and emphasize the importance of asset allocation. Avoiding the investor focus too much on the performance of the individual asset so that they can be more reasonable diversification of investment to reduce the risk. Thereby, optimizing the balance between investment return and risks.

Q3.1 Sharpe ratio

In [31]:
```
r_tg = (alpha * r_f - zeta)/(delta * r_f - alpha)
print(r_tg)
```

```
1.4862735358446917
```

In [32]:
```
sigama_tg = -((zeta - 2 * alpha * r_f + delta * (r_f**2)) **.5)/ (delta *
print(sigama_tg)
```

```
3.360726330566368
```

In [33]:
```
shar_ratio = (r_tg - r_f) / sigama_tg
print(shar_ratio)
```

```
0.40356559934950875
```

In [34]:
```
a = (delta * r_tg - alpha) / (zeta * delta - alpha ** 2)
```

In [35]:
```
b = (zeta - alpha * r_tg) / (zeta * delta - alpha ** 2)
```

In [36]:
```
w_star = a * np.dot(inv_V, R) + b * np.dot(inv_V, e)
print(w_star)
```

```
[ 0.56797218 -0.2140726   0.71410511  0.10408719 -0.36343817 -0.09546326
  0.99164683  0.0755702   0.13264333 -0.91305081]
```

In [37]:
```
df = pd.DataFrame(w_star, data_1.columns, columns = ['WEIGHT'])
print(df)
```

```
         WEIGHT
NoDur  0.567972
Durbl -0.214073
Manuf  0.714105
Enrgy  0.104087
HiTec -0.363438
Telcm -0.095463
Shops  0.991647
Hlth   0.075570
Utils  0.132643
Other -0.913051
```

Q3.2 EXPLANATION

The tangency portfolio helps investors understand the tradeoff between the risk and the return. Then, investors can evaluate a portfolio of different risks to make better investment strategies. In the equilibrium market, the optimal risk-return portfolio of investment in the CML. The bigger the Sharpe ratio, the better the portfolio, the bigger the expected return.