QF600 Asset Pricing Homework 4

Efficient Frontier Revisited

Wang Hairuo

2022-10-10

Contents

Part 1: Minimum-Track-Error Frontier

Let the market return be the target return.

Question 1: Estimate the Expected Deviation from Market Return for the Ten Industry Portfolios. Also Estimate the Covariance Matrix of Return Deviation for Ten Industry Portfolios.

Table 1: Expected Deviation from Market Return for Ten Industry Portfolios

	Expected Return Deviation
NoDur	0.1547
Durbl	-0.0147
Manuf	0.2648
Enrgy	0.4831
HiTec	0.0182
Telcm	0.1333
Shops	0.1683
Hlth	0.0358
Utils	0.1591
Other	-0.2590

According to **Table 1**, which shows the expected return deviation from market return for ten industry portfolios.

Table 2: The Covariance Matrix of Return Deviation for Ten Industry Portfolios

	NoDur	Durbl	Manuf	Enrgy	HiTec	Telcm	Shops	Hlth	Utils	Other
NoDur	5.440	-6.073	-1.396	-1.201	-1.883	1.539	1.141	3.815	4.272	-1.769
Durbl	-6.073	26.629	4.908	-3.481	1.892	-1.708	-0.354	-8.083	-9.617	4.386
Manuf	-1.396	4.908	2.950	1.666	0.065	-0.626	-1.155	-2.289	-1.901	0.359
Enrgy	-1.201	-3.481	1.666	19.275	-1.517	-1.041	-3.710	-2.486	4.454	-3.865
HiTec	-1.883	1.892	0.065	-1.517	5.099	-0.773	-0.245	-1.936	-2.343	-1.404
Telcm	1.539	-1.708	-0.626	-1.041	-0.773	4.683	0.464	0.693	2.721	-1.272
Shops	1.141	-0.354	-1.155	-3.710	-0.245	0.464	4.453	0.765	-0.177	-0.257
Hlth	3.815	-8.083	-2.289	-2.486	-1.936	0.693	0.765	7.820	3.496	-1.727
Utils	4.272	-9.617	-1.901	4.454	-2.343	2.721	-0.177	3.496	12.267	-4.055
Other	-1.769	4.386	0.359	-3.865	-1.404	-1.272	-0.257	-1.727	-4.055	4.503

Table 2 shows the covariance matrix of return deviation for ten industry portfolios.

Question 2: Plot the minimum-tracking-error frontier generated by the ten industry portfolios, with expected (monthly) return deviation on the vertical axis and (monthly) tracking error on the horizontal axis. This plot should cover the range from 0% to 0.1% on the vertical axis, in increments of 0.005% (or less).

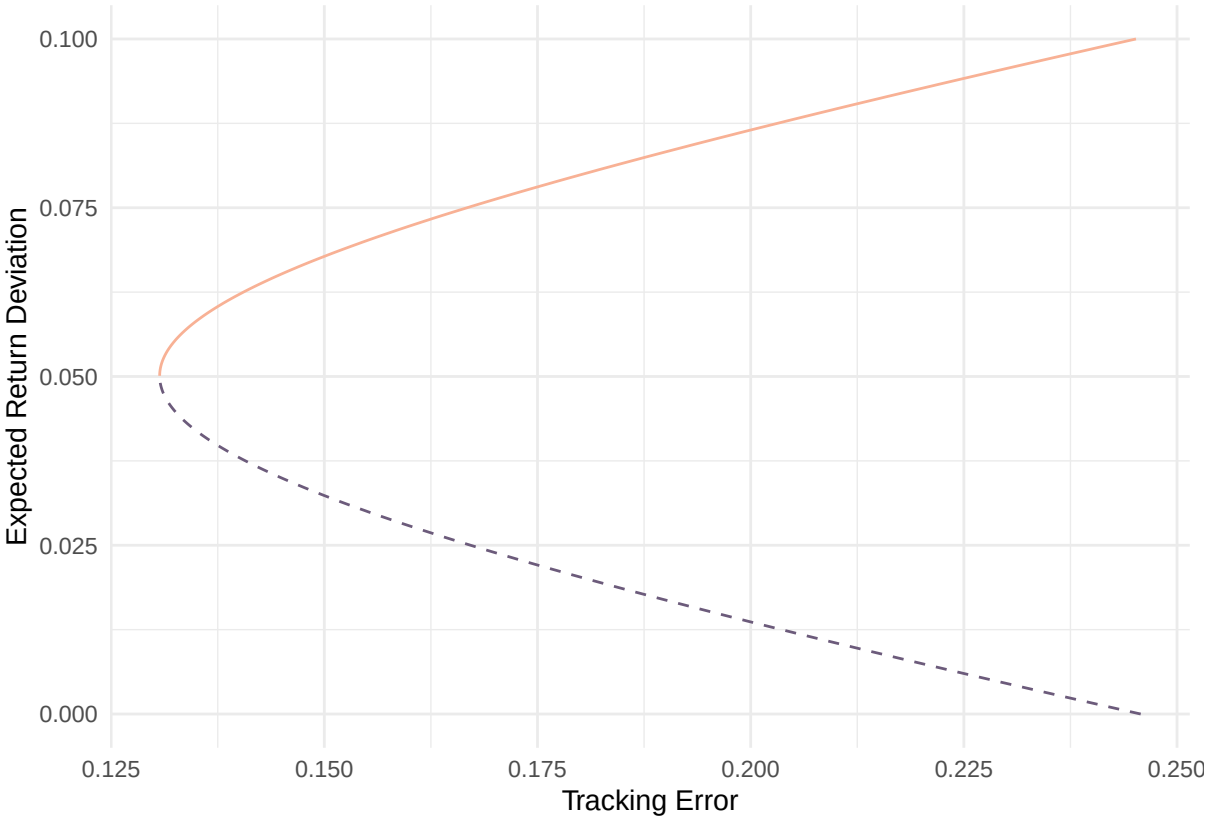

Figure 1: The Minimum-Tracking-Error Frontier

Figure 1 shows the minimum-tracking-error frontier which generated by ten industry portfolios.

Question 2: Also plot the line starting from the origin that is tangent to the upper half of the minimum-tracking-error frontier, and calculate the information ratio and portfolio weights for the "tangency" portfolio.

Plot of Minimum-Tracking-Error Frontier and Tangent Line to Upper Frontier

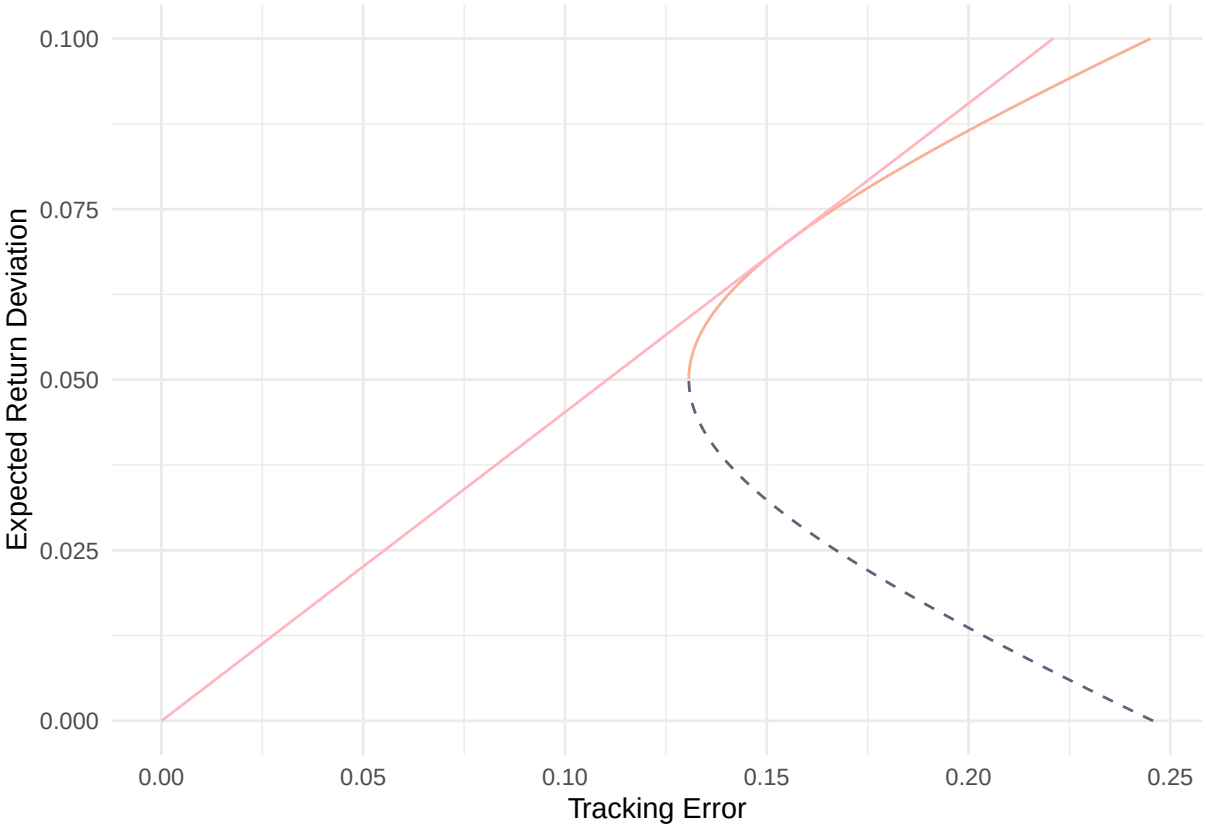

Figure 2: Minimum-Tracking-Error Frontier and Tangent Line for Upper Frontier

Figure 2 shows the minimum-tracking-error frontier and the tangent line to the frontier. There is a tangent point on the plot, and that represents the tangency portfolio.

Information Ratio and Portfolio Weights for "Tangency" Portfolio

$$I_i = \frac{E(R_i - R_m)}{\sqrt{Var(R_i - R_m)}}$$

This is the formula for the Information Ratio, which represent the expected deviation from market return, per unit of tracking error.

As we generated the scalar α, ζ and δ of Lagrange multipliers by the deviation of industry portfolio return from market rate. Therefore, we could compute the expected deviation of tangency portfolio by the scalar of ζ and α and with following formula,

$$E(R_{tg} - R_m) = \frac{\zeta}{\alpha}$$

Moreover, the expected deviation for tangency portfolio is 0.0698

As well, we could compute the tracking error for tangency portfolio by the following formula and consider risk-free rate as 0,

$$\sqrt{Var(R_{tg} - R_m)} = -\frac{\zeta - 2\alpha R_f + \delta R_f^2}{\delta(R_f - R_{mv})}$$

As we consider risk-free rate as 0, and the tracking error is:

$$\sqrt{Var(R_{tg} - R_m)} = -\frac{\zeta}{\delta(-R_{mv})}$$

Therefore, the tracking error for tangency portfolio is 0.1543

Overall, we could compute the information ratio by above formula and **the information ratio for the tangency portfolio in this case is 0.4525**

Table 3: The Portfolio Weights for the Tangency Portfolio

	Portfolio Weights
NoDur	0.0526342
Durbl	0.0001534
Manuf	0.1376267
Enrgy	0.0870322
HiTec	0.1793529
Telcm	0.0710740
Shops	0.1068845
Hlth	0.1027760
Utils	0.0401622
Other	0.2223039

According to **Table 3**, which is the each industry portfolio weights for the tangency portfolio. The positive portfolio weight represents the normal investment or long position.

Part 2: Minimum-Variance Frontier

Use the monthly returns of the ten industry portfolios to generate the minimum-variance frontier without short sales, using Monte Carlo simulation. Portfolio weights will be limited to the range [0, 1]. Randomly draw each element of w, the vector of portfolio weights, from the (standard) uniform distribution in the range [0, 1]. Divide w by the sum of the portfolio weights, to ensure that the portfolio weights sum to one. Use the normalized w to calculate the mean return and standard deviation of return for the simulated portfolio. Repeat this process until you have (at least) 10^5 data points.

Question 1: Plot the data points with mean return on the vertical axis and standard deviation of return on the horizontal axis.

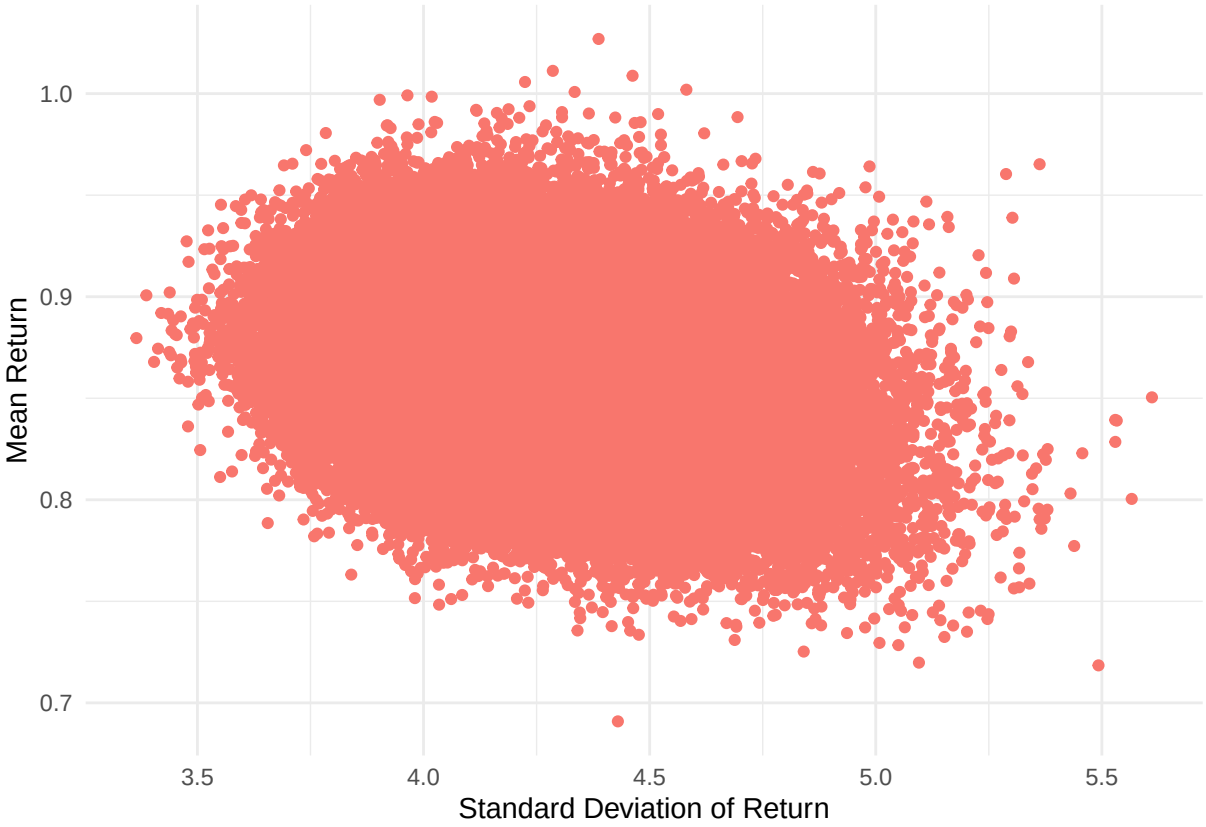

Figure 3: Minimum-Variance Frontier with Simulated Data Points

Figure 3 shows the minimum-variance frontier based on the simulated weight from standard uniform distribution.

Repeat this entire process by simulating 1/w using the standard uniform distribution, then take the reciprocal of the random draw from the standard uniform distribution as the portfolio weight.

Question 2: Plot the new data points with mean return on the vertical axis and standard deviation of return on the horizontal axis.

Figure 4: Minimum-Variance Frontier with Simulated Data Points

Figure 4 shows the minimum-variance frontier based on the simulated reciprocal of weight from standard uniform distribution.

Appendix

Set up Code

```
library(tidyverse)
library(ggplot2)
library(knitr)
library(readxl)
```

```
industry_df = read_excel("Industry_Portfolios.xlsx")
industry_df = industry_df %>%
  select(!Date)
market_df = read_excel("Market_Portfolio.xlsx")
market_df = market_df %>%
  select(!Date)
```

Part 1 - Question 1

```
# compute the return deviation
new_industry_df = industry_df - market_df$Market
# expected return deviation
mean_return = colMeans(new_industry_df)
# create a table to show the expected return deviation
kable(mean_return,
      col.names = "Expected Return Deviation",
      caption = "Expected Deviation from Market Return for Ten Industry Portfolios",
      digits = 4)
```

```
# covariance matrix for return deviation
cov_matrix = cov(new_industry_df)
# create a table to show the covariance matrix
kable(cov_matrix,
      caption = "The Covariance Matrix of Return Deviation for Ten Industry
      ↪  Portfolios",
      digits = 3)
```

Part 1 - Question 2

```
# compute the return deviation
new_industry_df = industry_df - market_df$Market
# expected return deviation
mean_return = colMeans(new_industry_df)
# the covariance matrix for return deviation
cov_matrix = cov(new_industry_df)
# the inverse covariance matrix for return deviation
inverse_cov_matrix = solve(cov_matrix)
# the number of industry portfolios
n = ncol(new_industry_df)
# unit vector
e_vector = as.vector(rep(x = 1, times = n))
```

```
# alpha
alpha = t(mean_return) %*% inverse_cov_matrix %*% e_vector
# zeta
zeta = t(mean_return) %*% inverse_cov_matrix %*% mean_return
# delta
delta = t(e_vector) %*% inverse_cov_matrix %*% e_vector
```

```
# set the given expected return deviation
return_p = seq(0, 0.1, 0.0001)
# compute the mean return for global minimum-tracking-error portfolio
return_mv = alpha / delta
# compute the tracking error with given expected return deviation
sigma_p = sqrt((1 / as.numeric(delta)) + (as.numeric(delta) / (as.numeric(zeta) *
↪  as.numeric(delta) - as.numeric(alpha) ^ 2)) * (return_p - as.numeric(return_mv)) ^
↪  2)
# record the expected return deviation and tracking error into data frame
plot_df = data.frame(return_p, sigma_p)
```

```
# plot the minimum-tracking-error frontier
plot_df %>%
  # select tracking error as x-axis and expected return deviation as y-axis
  ggplot(aes(x = sigma_p, y = return_p)) +
  # plot the top half of frontier as solid line
  geom_path(data = subset(plot_df, return_p >= as.numeric(return_mv)), linetype = 1,
  ↪  color = '#F8B195')+
```

```
  # plot the bottom half of frontier as dotted line
  geom_path(data = subset(plot_df, return_p < as.numeric(return_mv)), linetype = 2,
  ↪  color = '#6C5B7B')+
  # use the minimalstic theme for the plot
  theme_minimal() +
  # add x label
  xlab("Tracking Error") +
  # add y label
  ylab("Expected Return Deviation")
```

Part 1 - Question 3

```
# compute the tracking error with risk-free rate of zero
sigma_p2 = sqrt(((return_p ** 2) / as.numeric(zeta)))
# add the tracking error for tangent line into the data frame
plot_df2 = data.frame(return_p, sigma_p, sigma_p2)
```

```
# plot the minimum tracking error frontier and the tangent line for the frontier
plot_df2 %>%
  # use tracking error as x-axis and return deviation as y-axis
  ggplot(aes(x = sigma_p, y = return_p)) +
  # plot the top half of frontier with solid line
  geom_path(data = subset(plot_df2, return_p >= as.numeric(return_mv)), linetype = 1,
  ↪  color = '#F8B195') +
  # plot the bottom half of frontier with dotted line
  geom_path(data = subset(plot_df2, return_p < as.numeric(return_mv)), linetype = 2,
  ↪  color = '#6C5B7B')+
  # plot the tangent line to the frontier
  geom_path(aes(x = sigma_p2, y = return_p), color = "light pink") +
  # use the minimalistic theme to the plot
  theme_minimal() +
  # add x label
  xlab("Tracking Error") +
  # add y label
  ylab("Expected Return Deviation")
```

```
# Information Ratio
# return deviation for tangency portfolio
return_tg = - as.numeric(zeta) / - as.numeric(alpha)
```

```
# tracking error for tangency portfolio
sigma_tg = - ((as.numeric(zeta)) ^ 0.5) / (as.numeric(delta) * (-return_mv))
# compute the information ratio
information_ratio = return_tg / sigma_tg
```

```
a = ((as.numeric(zeta) * inverse_cov_matrix %*% e_vector) - (as.numeric(alpha) *
↪  inverse_cov_matrix %*% mean_return)) / (as.numeric(zeta) * as.numeric(delta) -
↪  as.numeric(alpha) ^ 2)

b = ((as.numeric(delta) * inverse_cov_matrix %*% mean_return) - (as.numeric(alpha) *
↪  inverse_cov_matrix %*% e_vector)) / (as.numeric(zeta) * as.numeric(delta) -
↪  as.numeric(alpha) ^ 2)

tangency_weight = a + b * return_tg
kable(tangency_weight,
      col.names = "Portfolio Weights",
      caption = "The Portfolio Weights for the Tangency Portfolio")
```

Part 2 - Question 1

```
# set seed in order to get same random weight
set.seed(600)
```

```
# mean return for industry portfolios
mean_return2 = colMeans(industry_df)
# covariance matrix for industry portfolios
cov_matrix2 = cov(industry_df)
```

```
# number of industry portfolio that we simulate
number_asset = ncol(industry_df)
# number of simulated data points
number_weights = 10^5

# create list and data frame to record the simulated mean return and standard
↪  deviation of return
weight_df = list()
sim_return_df = data.frame()
sim_sigma_df = data.frame()
```

```
# first set for simulating weights and compute the simulated mean return and sd of
↪  return
for (i in 1:number_weights / 2){
  # randomly draw weight from uniform distribution
  weights = runif(number_asset, 0, 1)
  # normalized the weights
  weights = weights / sum(weights)
  returns = mean_return2 %*% weights
  sigma = sqrt(t(weights) %*% cov_matrix2 %*% weights)
  weight_df[i] = list(weights)
  sim_return_df[i, 1] = as.numeric(returns)
  sim_sigma_df[i, 1] = as.numeric(sigma)
}
```

```
# second set for simulating weights and compute the simulated mean return and sd of
↪  return
for (i in (number_weights / 2 + 1):number_weights){
  # randomly draw weights from uniform distribution
  weights = runif(number_asset, 0, 1)
  # normalized the weights
  weights = weights / sum(weights)
  returns = mean_return2 %*% weights
  sigma = sqrt(t(weights) %*% cov_matrix2 %*% weights)
  weight_df[i] = list(weights)
  sim_return_df[i, 1] = as.numeric(returns)
  sim_sigma_df[i, 1] = as.numeric(sigma)
}
```

```
# put simulated mean return and standard deviation of mean into data frame
plot_df3 = data.frame(sim_return_df, sim_sigma_df)
# change column name for the data frame
colnames(plot_df3) = c("return", "sd")
```

```
# plot the minimum-variance frontier with simulated data points
plot_df3 %>%
  # select standard deviation of return on x-axis and mean return on y-axis
  ggplot(aes(x = sd, y = return)) +
  # set the color the points
  geom_point(aes(color = '#C06C84')) +
```

```
  # use minimalistic theme for the plots
  theme_minimal() +
  # add x label
  xlab("Standard Deviation of Return") +
  # add y label
  ylab("Mean Return")+
  # remove the non-informative legend
  theme(legend.position = 'None')
```

Part 2 - Question 2

```
# number of industry portfolios that we simulate
number_asset = ncol(industry_df)
# number of data point that we are going to simulate
number_weights = 10^5

# create list and data frame to save the simulated mean return and standard deviation
↪  of return
weight_df2 = list()
sim_return_df2 = data.frame()
sim_sigma_df2 = data.frame()

# first set for simulating weights and compute the simulated mean return and standard
↪  deviation of return
for (i in 1:number_weights / 2){
  # randomly draw 1/weight from uniform distribution
  weights = runif(number_asset, 0, 1)
  # compute the reciprocal of 1/weights
  weights = 1 / weights
  # normalized the weights
  real_weight = weights / sum(weights)
  returns = mean_return2 %*% real_weight
  sigma = sqrt(t(real_weight) %*% cov_matrix2 %*% real_weight)
  weight_df2[i] = list(real_weight)
  sim_return_df2[i, 1] = as.numeric(returns)
  sim_sigma_df2[i, 1] = as.numeric(sigma)
}
```

```
# second set for simulating weights and compute the simulated mean return and standard
↪  deviation of return
for (i in (number_weights / 2 + 1): number_weights){
  # randomly draw 1/weight from uniform distribution
  weights = runif(number_asset, 0, 1)
  # compute the reciprocal of 1/weights
  weights = 1 / weights
  # normalized the weights
  real_weight = weights / sum(weights)
  returns = mean_return2 %*% real_weight
  sigma = sqrt(t(real_weight) %*% cov_matrix2 %*% real_weight)
  weight_df2[i] = list(real_weight)
  sim_return_df2[i, 1] = as.numeric(returns)
  sim_sigma_df2[i, 1] = as.numeric(sigma)
}
```

```
# put simulated mean return and standard deviation of mean into data frame
plot_df4 = data.frame(sim_return_df2, sim_sigma_df2)
# change column names
colnames(plot_df4) = c("return", "sd")
```

```
# plot the minimum-variance frontier with simulated data points
plot_df4 %>%
  # select standard deviation of return as x-axis and mean return as y-axis
  ggplot(aes(x = sd, y = return)) +
  # set the color for data points
  geom_point(aes(color = '#F8B195')) +
  # use minimalistic theme
  theme_minimal() +
  # add x label
  xlab("Standard Deviation of Return") +
  # add y label
  ylab("Mean Return") +
  # remove the non-informative legend
  theme(legend.position = 'None')
```