In [54]:
```
import pandas as pd
import numpy as np
from statsmodels.regression.rolling import RollingOLS
from scipy import stats
import statsmodels.regression.linear_model as sm
import statsmodels.tools.tools as sm2
from statsmodels.regression.linear_model import OLS
#from pyfinance import PandasRollingOLS
from statsmodels.api import add_constant
```

In [55]:
```
#read in ffdata
new_row = pd.DataFrame({'-0.000800':-0.000800, '0.008300':0.008300, '0.0
04100':0.004100,
                        '0.000040':0.000040}, index = [0])
ffdata = pd.read_csv("ffdata.txt", delim_whitespace=True)
ffdata = pd.concat([new_row, ffdata]).reset_index(drop = True)
ffdata.rename(columns={'-0.000800':'Market Returns', '0.008300':'Returns
to the Fama-French size factor',
                       '0.004100':'Returns to the Fama-French value fact
or',
                       '0.000040':'Risk-free rate'}, inplace = True)
ffdata
```

Out[55]:

	Market Returns	Returns to the Fama-French size factor	Returns to the Fama-French value factor	Risk-free rate
0	-0.0008	0.0083	0.0041	0.00004
1	0.0122	0.0035	0.0002	0.00004
2	0.0019	0.0012	0.0018	0.00004
3	0.0027	0.0050	-0.0007	0.00004
4	0.0048	0.0033	0.0062	0.00004
...	...	...	...	...
1506	0.0052	-0.0008	0.0016	0.00000
1507	0.0010	-0.0015	-0.0030	0.00000
1508	-0.0009	0.0008	-0.0008	0.00000
1509	-0.0007	-0.0004	-0.0018	0.00000
1510	-0.0088	-0.0013	0.0015	0.00000

1511 rows × 4 columns

In [56]:
```
#load in ticker.txt - tickers
ticker = pd.read_csv('ticker.txt', header=None)
ticker = ticker.reset_index()
ticker['index'] = ticker['index'] + 1
ticker
```

Out[56]:

	index	0
0	1	A
1	2	AA
2	3	AAI
3	4	AAON
4	5	AAP
...	...	...
1872	1873	ZMH
1873	1874	ZOLL
1874	1875	ZOLT
1875	1876	ZQK
1876	1877	ZRAN

1877 rows × 2 columns

In [204]:
```
#load in retdate.txt - return dates
retdate = pd.read_csv('retdate.txt', sep = " ", header = None)
retdate.loc[1510]
```

Out[204]:
```
0    20091231
Name: 1510, dtype: int64
```

In [58]:
```
#load in secdata.txt - securities data
secdata = pd.read_csv('secdata.txt', sep = " ", header = None)
secdata.columns = ['Ticker #', 'Stock Returns', 'Market Capitalizations'
]
secdata
```

Out[58]:

	Ticker #	Stock Returns	Market Capitalizations
0	1	-0.015048	13713091.20
1	1	0.026042	14070202.95
2	1	0.031472	14513021.52
3	1	0.012795	14698719.63
4	1	0.045675	15370089.72
...	...	...	...
2836142	1877	0.000000	580951.00
2836143	1877	-0.008079	576257.50
2836144	1877	0.000000	576257.50
2836145	1877	0.008145	580951.00
2836146	1877	-0.008079	576257.50

2836147 rows × 3 columns

In [59]:
```
#merge ticker with stocks
stock_with_ticker = secdata.merge(ticker, left_on = "Ticker #", right_on
= "index")
stock_with_ticker.to_csv('stock_with_ticker.csv')
```

Question 2

We now consider the modern portfolio theory (MPT) approach to estimating volatility. Each step below should be completed using 504, 252, 126, and 63 day rolling windows.

a) Pick a portfolio of 100 securities.

Criteria: 1st 100 securities.

In [60]:
```
secdata
```

Out[60]:

	Ticker #	Stock Returns	Market Capitalizations
0	1	-0.015048	13713091.20
1	1	0.026042	14070202.95
2	1	0.031472	14513021.52
3	1	0.012795	14698719.63
4	1	0.045675	15370089.72
...	...	...	...
2836142	1877	0.000000	580951.00
2836143	1877	-0.008079	576257.50
2836144	1877	0.000000	576257.50
2836145	1877	0.008145	580951.00
2836146	1877	-0.008079	576257.50

2836147 rows × 3 columns

In [61]:
```
random_tickers = list(range(1,101))
```

In [62]:
```
#turn tickers into columns
secdata_group = secdata.set_index([secdata.groupby('Ticker #').cumcount
(), 'Ticker #'])['Stock Returns'].unstack()
secdata_group
```

Out[62]:

Ticker #	1	2	3	4	5	6	7	8	
0	-0.015048	-0.011842	0.030252	-0.024729	0.000246	-0.004212	-0.072416	-0.038290	0.0
1	0.026042	0.032756	0.008157	0.004754	-0.009089	0.041823	-0.072706	0.033889	0.0
2	0.031472	-0.007478	0.062298	0.014196	0.017848	-0.003608	0.097044	-0.004120	0.0
3	0.012795	-0.007534	0.022848	-0.005184	0.031417	0.022635	0.062683	0.009712	0.0
4	0.045675	0.012042	-0.067014	0.001042	-0.026446	0.034086	0.006064	-0.004809	0.0
...	...	...	...	...	...	...	...	...	
1506	0.000990	0.021250	0.009363	0.011190	0.000242	0.034339	0.019135	0.000760	0.0
1507	0.002308	-0.014688	-0.035250	-0.010060	-0.003867	0.012294	0.016327	0.002658	0.0
1508	-0.002303	-0.004348	-0.007692	0.013720	-0.011160	-0.011861	0.012450	-0.001515	0.0
1509	0.025387	0.016843	0.007752	-0.005013	0.006133	0.012147	0.015470	-0.002275	0.0
1510	-0.000965	-0.011043	0.003846	-0.018136	-0.012924	-0.004290	-0.001953	-0.009122	0.0

1511 rows × 1877 columns

In [63]:
```
portfolio_q2_ret = secdata_group[random_tickers]
portfolio_q2_ret
```

Out[63]:

Ticker #	1	2	3	4	5	6	7	8	
0	-0.015048	-0.011842	0.030252	-0.024729	0.000246	-0.004212	-0.072416	-0.038290	0.0
1	0.026042	0.032756	0.008157	0.004754	-0.009089	0.041823	-0.072706	0.033889	0.0
2	0.031472	-0.007478	0.062298	0.014196	0.017848	-0.003608	0.097044	-0.004120	0.0
3	0.012795	-0.007534	0.022848	-0.005184	0.031417	0.022635	0.062683	0.009712	0.0
4	0.045675	0.012042	-0.067014	0.001042	-0.026446	0.034086	0.006064	-0.004809	0.0
...	...	...	...	...	...	...	...	...	
1506	0.000990	0.021250	0.009363	0.011190	0.000242	0.034339	0.019135	0.000760	0.0
1507	0.002308	-0.014688	-0.035250	-0.010060	-0.003867	0.012294	0.016327	0.002658	0.0
1508	-0.002303	-0.004348	-0.007692	0.013720	-0.011160	-0.011861	0.012450	-0.001515	0.0
1509	0.025387	0.016843	0.007752	-0.005013	0.006133	0.012147	0.015470	-0.002275	0.0
1510	-0.000965	-0.011043	0.003846	-0.018136	-0.012924	-0.004290	-0.001953	-0.009122	0.0

1511 rows × 100 columns

In [64]:
```
#market cap grouped by ticker #
secdata_cap_group = secdata.set_index([secdata.groupby('Ticker #').cumcount(),
                                       'Ticker #'])['Market Capitalizations'].unstack()
secdata_cap_group
```

Out[64]:

Ticker #	1	2	3	4	5	6	7	
0	13713091.20	32494080.25	1031397.02	237003.60	3004886.52	7.847234e+06	325464.88	6
1	14070202.95	33558466.90	1039809.72	238130.40	2977576.08	8.175431e+06	301801.76	6
2	14513021.52	33307513.95	1104587.51	241510.80	3030720.72	8.145930e+06	331089.72	6
3	14698719.63	33056561.00	1129825.61	240258.80	3125938.20	8.330311e+06	351843.44	6
4	15370089.72	33454624.30	1054111.31	240509.20	3043268.76	8.614257e+06	353977.00	6
...	...	...	...	...	...	...	...	
1506	10467246.96	15921336.52	725246.06	342293.84	3915830.78	1.882777e+08	539539.00	7
1507	10491404.80	15687485.80	699680.80	338850.24	3900689.82	1.905925e+08	548347.80	7
1508	10467246.96	15619279.34	694298.64	343499.10	3857159.56	1.883318e+08	555174.62	7
1509	10732983.20	15882361.40	699680.80	341777.30	3880817.31	1.906195e+08	563763.20	7
1510	10838179.17	15706973.36	702371.88	335578.82	3830662.88	1.898017e+08	562662.10	7

1511 rows × 1877 columns

In [65]:
```
portfolio_q2_cap = secdata_cap_group[random_tickers]
portfolio_q2_cap
```

Out[65]:

Ticker #	1	2	3	4	5	6	7	
0	13713091.20	32494080.25	1031397.02	237003.60	3004886.52	7.847234e+06	325464.88	6
1	14070202.95	33558466.90	1039809.72	238130.40	2977576.08	8.175431e+06	301801.76	6
2	14513021.52	33307513.95	1104587.51	241510.80	3030720.72	8.145930e+06	331089.72	6
3	14698719.63	33056561.00	1129825.61	240258.80	3125938.20	8.330311e+06	351843.44	6
4	15370089.72	33454624.30	1054111.31	240509.20	3043268.76	8.614257e+06	353977.00	6
...	...	...	...	...	...	...	...	
1506	10467246.96	15921336.52	725246.06	342293.84	3915830.78	1.882777e+08	539539.00	7
1507	10491404.80	15687485.80	699680.80	338850.24	3900689.82	1.905925e+08	548347.80	7
1508	10467246.96	15619279.34	694298.64	343499.10	3857159.56	1.883318e+08	555174.62	7
1509	10732983.20	15882361.40	699680.80	341777.30	3880817.31	1.906195e+08	563763.20	7
1510	10838179.17	15706973.36	702371.88	335578.82	3830662.88	1.898017e+08	562662.10	7

1511 rows × 100 columns

For part b:

Standard deviation of portfolio = portfolio volatility.

Equation:

$$\hat{\sigma}_{Portfolio} = \sqrt{w_T \cdot \Sigma \cdot w}$$

where:

- w is portfolio weights
- Σ is covariance matrix
- $\cdot$ the dot-multiplication for matrix multiplication
- $\hat{\sigma}_{Portfolio}$ is the estimated portfolio volatility/standard deviation

In [66]:
```
#function to find portfolio standard deviation
def sd_portfolio(cov_mat, arr_weights):
    if np.isnan(cov_mat).any():
        return cov_mat
    return np.dot(np.dot(np.transpose(arr_weights), cov_mat), arr_weight
s)**0.5
```

Rolling Window 504

i) Generate a covariance matrixes for generated portfolio.

In [67]:
```
cov_df_504 = portfolio_q2_ret.rolling(504).cov()
cov_df_504.dropna(inplace = True)
cov_df_504.drop(cov_df_504.tail(100).index, inplace = True)
cov_df_504_np = cov_df_504.to_numpy()
cov_df_504
```

Out[67]:

	Ticker #	1	2	3	4	5	6	7	8	
	Ticker #									
503	1	0.000438	0.000085	0.000203	0.000117	0.000071	0.000140	0.000142	0.000058	0.
	2	0.000085	0.000255	0.000123	0.000068	0.000062	0.000122	0.000086	0.000053	0.
	3	0.000203	0.000123	0.000910	0.000135	0.000131	0.000171	0.000280	0.000100	0.
	4	0.000117	0.000068	0.000135	0.000482	0.000065	0.000090	0.000122	0.000040	0.
	5	0.000071	0.000062	0.000131	0.000065	0.000316	0.000070	0.000099	0.000007	0.
...	...	...	...	...	...	...	...	...	...	
1509	96	0.000409	0.000568	0.000574	0.000460	0.000312	0.000247	0.000448	0.000218	0.
	97	0.000689	0.001155	0.001032	0.000897	0.000628	0.000641	0.000685	0.000323	0.
	98	0.000728	0.000893	0.002917	0.001070	0.000794	0.000664	0.000785	0.000371	0.
	99	0.000591	0.000803	0.000918	0.000784	0.000431	0.000454	0.000610	0.000323	0.
	100	0.000938	0.001578	0.001011	0.000973	0.000742	0.000784	0.000960	0.000419	0.

100700 rows × 100 columns

ii) Estimate the standard deviations of the portfolio over rw (from the last day in the rolling window).

In [68]:
```
#portfolio_weights
weights_q2_504 = portfolio_q2_cap.iloc[:, :].apply(lambda x: x.div(x.sum
()), axis=1)
df_weights_q2_504 = weights_q2_504.drop(weights_q2_504.tail(1).index)
arr_weights_q2_504 = df_weights_q2_504.tail(1007).to_numpy()
df_weights_q2_504
```

Out[68]:

Ticker #	1	2	3	4	5	6	7	8	9
0	0.018568	0.043998	0.001397	0.000321	0.004069	0.010625	0.000441	0.008199	0.000216
1	0.018809	0.044861	0.001390	0.000318	0.003980	0.010929	0.000403	0.008369	0.000217
2	0.019375	0.044465	0.001475	0.000322	0.004046	0.010875	0.000442	0.008323	0.000218
3	0.019478	0.043805	0.001497	0.000318	0.004142	0.011039	0.000466	0.008342	0.000219
4	0.020277	0.044135	0.001391	0.000317	0.004015	0.011364	0.000467	0.008265	0.000219
...	...	...	...	...	...	...	...	...	...
1505	0.013604	0.020282	0.000935	0.000440	0.005093	0.236809	0.000689	0.009864	0.000120
1506	0.013457	0.020469	0.000932	0.000440	0.005034	0.242061	0.000694	0.009756	0.000119
1507	0.013420	0.020067	0.000895	0.000433	0.004990	0.243796	0.000701	0.009732	0.000123
1508	0.013447	0.020065	0.000892	0.000441	0.004955	0.241941	0.000713	0.009759	0.000125
1509	0.013719	0.020300	0.000894	0.000437	0.004960	0.243645	0.000721	0.009688	0.000126

1510 rows × 100 columns

In [69]:
```
sd_port_504 = []
for i in range(0, len(cov_df_504_np), 100):
    idx = 0
    sd_port_504.append(sd_portfolio(cov_df_504_np[i:i+100], arr_weights_
q2_504[idx]))
    idx += 1
sd_port_504_arr = np.array(sd_port_504)
sd_port_504_arr
```

Out[69]:
```
array([0.00820984, 0.00824556, 0.00823351, ..., 0.03161844, 0.03161853,
       0.03161824])
```

iii) Using the market capitalization weights and returns (from the day following the last day in the rolling window) of your securities, calculate the one-day ahead return of the portfolio, $\tilde{r}_p$.

In [70]:
```
#getting one-day ahead returns array
dayahead504_port_ret_q2 = []
dayahead504_ret_q2 = portfolio_q2_ret.loc[504:1510].to_numpy()
dayahead504_w_q2 = weights_q2_504.loc[504:1510].to_numpy()
for i in range(0, len(dayahead504_w_q2)):
    dayahead504_port_ret_q2.append(np.multiply(dayahead504_ret_q2[i], da
yahead504_w_q2[i]))
dayahead504_port_ret_q2_arr = np.sum(np.array(dayahead504_port_ret_q2),
axis = 1)
dayahead504_port_ret_q2_arr
```

Out[70]:
```
array([ 0.01826034,  0.00564215, -0.00055066, ..., -0.00414611,
        0.00514476, -0.00946444])
```

iv) Calculate the standardized outcome, $\tilde{z}_p$, where $\tilde{z}_p = \frac{\tilde{r}_p}{\hat{\sigma}_p}$ where we make the simplifying assumption that $E[\tilde{r}_p] = 0$.

In [71]:
```
#dividing arrays to get standardized outcomes.
standardized_outcomes_504_q2 = dayahead504_port_ret_q2_arr / sd_port_504
_arr
std_outcomes_504_q2 = pd.DataFrame(standardized_outcomes_504_q2)
std_outcomes_504_q2.index += 504
std_outcomes_504_q2.rename(columns={0: "Standardized Outcome"}, inplace
= True)
std_outcomes_504_q2
```

Out[71]:

	Standardized Outcome
504	2.224203
505	0.684265
506	-0.066881
507	1.336911
508	0.451453
...	...
1506	0.382310
1507	0.164140
1508	-0.131129
1509	0.162713
1510	-0.299335

1007 rows × 1 columns

Rolling Window 252

i) Generate a covariance matrixes for generated portfolio.

In [72]:
```
cov_df_252 = portfolio_q2_ret.rolling(252).cov()
cov_df_252.dropna(inplace = True)
cov_df_252.drop(cov_df_252.tail(100).index, inplace = True)
cov_df_252_np = cov_df_252.to_numpy()
cov_df_252
```

Out[72]:

	Ticker #	1	2	3	4	5	6	7	8	
	Ticker #									
251	1	0.000583	0.000118	0.000287	0.000161	0.000108	0.000154	0.000203	0.000078	0.
	2	0.000118	0.000327	0.000146	0.000086	0.000071	0.000127	0.000116	0.000072	0.
	3	0.000287	0.000146	0.000980	0.000183	0.000155	0.000187	0.000392	0.000126	0.
	4	0.000161	0.000086	0.000183	0.000538	0.000077	0.000067	0.000177	0.000052	0.
	5	0.000108	0.000071	0.000155	0.000077	0.000305	0.000057	0.000108	0.000014	0.
...	...	...	...	...	...	...	...	...	...	
1509	96	0.000361	0.000421	0.000348	0.000372	0.000114	0.000126	0.000305	0.000147	0.
	97	0.000467	0.000730	0.000631	0.000512	0.000308	0.000345	0.000420	0.000180	0.
	98	0.000628	0.001012	0.001642	0.000609	0.000306	0.000381	0.000467	0.000249	0.
	99	0.000455	0.000677	0.000675	0.000526	0.000225	0.000251	0.000378	0.000226	0.
	100	0.000725	0.001194	0.000937	0.000610	0.000317	0.000482	0.000615	0.000217	0.

125900 rows × 100 columns

ii) Estimate the standard deviations of the portfolio over rw (from the last day in the rolling window).

In [73]:
```
#portfolio_weights
weights_q2_252 = portfolio_q2_cap.iloc[:, :].apply(lambda x: x.div(x.sum
()), axis=1)
df_weights_q2_252 = weights_q2_252.drop(weights_q2_252.tail(1).index)
arr_weights_q2_252 = df_weights_q2_252.tail(1259).to_numpy()
df_weights_q2_252
```

Out[73]:

Ticker #	1	2	3	4	5	6	7	8	9
0	0.018568	0.043998	0.001397	0.000321	0.004069	0.010625	0.000441	0.008199	0.000216
1	0.018809	0.044861	0.001390	0.000318	0.003980	0.010929	0.000403	0.008369	0.000217
2	0.019375	0.044465	0.001475	0.000322	0.004046	0.010875	0.000442	0.008323	0.000218
3	0.019478	0.043805	0.001497	0.000318	0.004142	0.011039	0.000466	0.008342	0.000219
4	0.020277	0.044135	0.001391	0.000317	0.004015	0.011364	0.000467	0.008265	0.000219
...	...	...	...	...	...	...	...	...	...
1505	0.013604	0.020282	0.000935	0.000440	0.005093	0.236809	0.000689	0.009864	0.000120
1506	0.013457	0.020469	0.000932	0.000440	0.005034	0.242061	0.000694	0.009756	0.000119
1507	0.013420	0.020067	0.000895	0.000433	0.004990	0.243796	0.000701	0.009732	0.000123
1508	0.013447	0.020065	0.000892	0.000441	0.004955	0.241941	0.000713	0.009759	0.000125
1509	0.013719	0.020300	0.000894	0.000437	0.004960	0.243645	0.000721	0.009688	0.000126

1510 rows × 100 columns

In [74]:
```
sd_port_252 = []
for i in range(0, len(cov_df_252_np), 100):
    idx = 0
    sd_port_252.append(sd_portfolio(cov_df_252_np[i:i+100], arr_weights_
q2_252[idx]))
    idx += 1
sd_port_252_arr = np.array(sd_port_252)
sd_port_252_arr
```

Out[74]:
```
array([0.00848636, 0.00849811, 0.00850478, ..., 0.03044609, 0.03043997,
       0.03041131])
```

iii) Using the market capitalization weights and returns (from the day following the last day in the rolling window) of your securities, calculate the one-day ahead return of the portfolio, $\tilde{r}_p$.

In [75]:
```
#getting one-day ahead returns array
dayahead252_port_ret_q2 = []
dayahead252_ret_q2 = portfolio_q2_ret.loc[252:1510].to_numpy()
dayahead252_w_q2 = weights_q2_252.loc[252:1510].to_numpy()
for i in range(0, len(dayahead252_w_q2)):
    dayahead252_port_ret_q2.append(np.multiply(dayahead252_ret_q2[i], da
yahead252_w_q2[i]))
dayahead252_port_ret_q2_arr = np.sum(np.array(dayahead252_port_ret_q2),
axis = 1)
dayahead252_port_ret_q2_arr
```

Out[75]:
```
array([-0.00643605, -0.01146078, -0.00250408, ..., -0.00414611,
        0.00514476, -0.00946444])
```

iv) Calculate the standardized outcome, $\tilde{z}_p$, where $\tilde{z}_p = \frac{\tilde{r}_p}{\hat{\sigma}_p}$ where we make the simplifying assumption that $E[\tilde{r}_p] = 0$.

In [76]:
```
standardized_outcomes_252_q2 = dayahead252_port_ret_q2_arr / sd_port_252
_arr
std_outcomes_252_q2 = pd.DataFrame(standardized_outcomes_252_q2)
std_outcomes_252_q2.index += 252
std_outcomes_252_q2.rename(columns={0: "Standardized Outcome"}, inplace
= True)
std_outcomes_252_q2
```

Out[76]:

	Standardized Outcome
252	-0.758400
253	-1.348626
254	-0.294432
255	0.541728
256	0.640736
...	...
1506	0.397123
1507	0.170497
1508	-0.136179
1509	0.169013
1510	-0.311215

1259 rows × 1 columns

Rolling Window 126

i) Generate a covariance matrixes for generated portfolio.

In [77]:
```
cov_df_126 = portfolio_q2_ret.rolling(126).cov()
cov_df_126.dropna(inplace = True)
cov_df_126.drop(cov_df_126.tail(100).index, inplace = True)
cov_df_126_np = cov_df_126.to_numpy()
cov_df_126
```

Out[77]:

	Ticker #	1	2	3	4	5	6	7	8	
	Ticker #									
125	1	0.000528	0.000192	0.000244	0.000163	0.000079	0.000183	0.000176	0.000096	0.
	2	0.000192	0.000408	0.000228	0.000095	0.000062	0.000162	0.000170	0.000098	0.
	3	0.000244	0.000228	0.001045	0.000181	0.000174	0.000204	0.000361	0.000105	0.
	4	0.000163	0.000095	0.000181	0.000595	0.000054	0.000046	0.000164	0.000057	0.
	5	0.000079	0.000062	0.000174	0.000054	0.000228	0.000089	0.000104	0.000034	0.
...	...	...	...	...	...	...	...	...	...	
1509	96	0.000177	0.000201	0.000162	0.000103	0.000020	0.000089	0.000153	0.000083	0.
	97	0.000233	0.000360	0.000289	0.000181	0.000154	0.000157	0.000230	0.000081	0.
	98	0.000241	0.000462	0.000875	0.000252	0.000170	0.000176	0.000315	0.000113	0.
	99	0.000180	0.000191	0.000192	0.000191	0.000055	0.000075	0.000078	0.000078	0.
	100	0.000318	0.000584	0.000173	0.000194	0.000197	0.000283	0.000283	0.000072	0.

138500 rows × 100 columns

ii) Estimate the standard deviations of the portfolio over rw (from the last day in the rolling window).

In [78]:
```
#portfolio_weights
weights_q2_126 = portfolio_q2_cap.iloc[:, :].apply(lambda x: x.div(x.sum
()), axis=1)
df_weights_q2_126 = weights_q2_126.drop(weights_q2_126.tail(1).index)
arr_weights_q2_126 = df_weights_q2_126.tail(1385).to_numpy()
df_weights_q2_126
```

Out[78]:

Ticker #	1	2	3	4	5	6	7	8	9
0	0.018568	0.043998	0.001397	0.000321	0.004069	0.010625	0.000441	0.008199	0.000216
1	0.018809	0.044861	0.001390	0.000318	0.003980	0.010929	0.000403	0.008369	0.000217
2	0.019375	0.044465	0.001475	0.000322	0.004046	0.010875	0.000442	0.008323	0.000218
3	0.019478	0.043805	0.001497	0.000318	0.004142	0.011039	0.000466	0.008342	0.000219
4	0.020277	0.044135	0.001391	0.000317	0.004015	0.011364	0.000467	0.008265	0.000219
...	...	...	...	...	...	...	...	...	...
1505	0.013604	0.020282	0.000935	0.000440	0.005093	0.236809	0.000689	0.009864	0.000120
1506	0.013457	0.020469	0.000932	0.000440	0.005034	0.242061	0.000694	0.009756	0.000119
1507	0.013420	0.020067	0.000895	0.000433	0.004990	0.243796	0.000701	0.009732	0.000123
1508	0.013447	0.020065	0.000892	0.000441	0.004955	0.241941	0.000713	0.009759	0.000125
1509	0.013719	0.020300	0.000894	0.000437	0.004960	0.243645	0.000721	0.009688	0.000126

1510 rows × 100 columns

In [79]:
```
sd_port_126 = []
for i in range(0, len(cov_df_126_np), 100):
    idx = 0
    sd_port_126.append(sd_portfolio(cov_df_126_np[i:i+100], arr_weights_
q2_126[idx]))
    idx += 1
sd_port_126_arr = np.array(sd_port_126)
sd_port_126_arr
```

Out[79]:
```
array([0.00840445, 0.0084857 , 0.00841868, ..., 0.02738999, 0.02715904,
       0.02676144])
```

iii) Using the market capitalization weights and returns (from the day following the last day in the rolling window) of your securities, calculate the one-day ahead return of the portfolio, $\tilde{r}_p$.

In [80]:
```
#getting one-day ahead returns array
dayahead126_port_ret_q2 = []
dayahead126_ret_q2 = portfolio_q2_ret.loc[126:1510].to_numpy()
dayahead126_w_q2 = weights_q2_126.loc[126:1510].to_numpy()
for i in range(0, len(dayahead126_w_q2)):
    dayahead126_port_ret_q2.append(np.multiply(dayahead126_ret_q2[i], da
yahead126_w_q2[i]))
dayahead126_port_ret_q2_arr = np.sum(np.array(dayahead126_port_ret_q2),
axis = 1)
dayahead126_port_ret_q2_arr
```

Out[80]:
```
array([-0.01262911,  0.00056176, -0.0061875 , ..., -0.00414611,
        0.00514476, -0.00946444])
```

iv) Calculate the standardized outcome, $\tilde{z}_p$, where $\tilde{z}_p = \frac{\tilde{r}_p}{\hat{\sigma}_p}$ where we make the simplifying assumption that $E[\tilde{r}_p] = 0$.

In [81]:
```
standardized_outcomes_126_q2 = dayahead126_port_ret_q2_arr / sd_port_126
_arr
std_outcomes_126_q2 = pd.DataFrame(standardized_outcomes_126_q2)
std_outcomes_126_q2.index += 126
std_outcomes_126_q2.rename(columns={0: "Standardized Outcome"}, inplace
= True)
std_outcomes_126_q2
```

Out[81]:

	Standardized Outcome
126	-1.502669
127	0.066201
128	-0.734973
129	-0.016849
130	0.009584
...	...
1506	0.440712
1507	0.189176
1508	-0.151373
1509	0.189431
1510	-0.353660

1385 rows × 1 columns

Rolling Window 63

i) Generate a covariance matrixes for generated portfolio.

In [82]:
```
cov_df_63 = portfolio_q2_ret.rolling(63).cov()
cov_df_63.dropna(inplace = True)
cov_df_63.drop(cov_df_63.tail(100).index, inplace = True)
cov_df_63_np = cov_df_63.to_numpy()
cov_df_63
```

Out[82]:

	Ticker #	1	2	3	4	5	6	7	8
	Ticker #								
62	1	0.000632	0.000198	0.000263	0.000097	0.000066	0.000207	0.000192	0.000119
	2	0.000198	0.000391	0.000206	-0.000027	0.000041	0.000140	0.000150	0.000113
	3	0.000263	0.000206	0.001327	0.000140	0.000238	0.000196	0.000387	0.000139
	4	0.000097	-0.000027	0.000140	0.000626	0.000007	-0.000044	0.000040	0.000023
	5	0.000066	0.000041	0.000238	0.000007	0.000237	0.000092	0.000085	0.000040
...	...	...	...	...	...	...	...	...	...
1509	96	0.000241	0.000316	0.000219	0.000169	0.000143	0.000141	0.000228	0.000099
	97	0.000248	0.000427	0.000427	0.000238	0.000290	0.000161	0.000354	0.000147
	98	0.000271	0.000581	0.001058	0.000278	0.000306	0.000132	0.000350	0.000154
	99	0.000113	0.000208	0.000064	0.000106	0.000098	0.000015	0.000145	0.000074
	100	0.000247	0.000608	0.000099	0.000176	0.000212	0.000252	0.000140	0.000189

144800 rows × 100 columns

ii) Estimate the standard deviations of the portfolio over rw (from the last day in the rolling window).

In [83]:
```
#portfolio_weights
weights_q2_63 = portfolio_q2_cap.iloc[:, :].apply(lambda x: x.div(x.sum
()), axis=1)
df_weights_q2_63 = weights_q2_63.drop(weights_q2_63.tail(1).index)
arr_weights_q2_63 = df_weights_q2_63.tail(1448).to_numpy()
df_weights_q2_63
```

Out[83]:

Ticker #	1	2	3	4	5	6	7	8	9
0	0.018568	0.043998	0.001397	0.000321	0.004069	0.010625	0.000441	0.008199	0.000216
1	0.018809	0.044861	0.001390	0.000318	0.003980	0.010929	0.000403	0.008369	0.000217
2	0.019375	0.044465	0.001475	0.000322	0.004046	0.010875	0.000442	0.008323	0.000218
3	0.019478	0.043805	0.001497	0.000318	0.004142	0.011039	0.000466	0.008342	0.000219
4	0.020277	0.044135	0.001391	0.000317	0.004015	0.011364	0.000467	0.008265	0.000219
...	...	...	...	...	...	...	...	...	...
1505	0.013604	0.020282	0.000935	0.000440	0.005093	0.236809	0.000689	0.009864	0.000120
1506	0.013457	0.020469	0.000932	0.000440	0.005034	0.242061	0.000694	0.009756	0.000119
1507	0.013420	0.020067	0.000895	0.000433	0.004990	0.243796	0.000701	0.009732	0.000123
1508	0.013447	0.020065	0.000892	0.000441	0.004955	0.241941	0.000713	0.009759	0.000125
1509	0.013719	0.020300	0.000894	0.000437	0.004960	0.243645	0.000721	0.009688	0.000126

1510 rows × 100 columns

In [84]:
```
sd_port_63 = []
for i in range(0, len(cov_df_63_np), 100):
    idx = 0
    sd_port_63.append(sd_portfolio(cov_df_63_np[i:i+100], arr_weights_q2
_63[idx]))
    idx += 1
sd_port_63_arr = np.array(sd_port_63)
sd_port_63_arr
```

Out[84]:
```
array([0.00885192, 0.00908816, 0.00906737, ..., 0.01689205, 0.01687095,
       0.01685115])
```

iii) Using the market capitalization weights and returns (from the day following the last day in the rolling window) of your securities, calculate the one-day ahead return of the portfolio, $\tilde{r}_p$.

In [85]:
```
#getting one-day ahead returns array
dayahead63_port_ret_q2 = []
dayahead63_ret_q2 = portfolio_q2_ret.loc[63:1510].to_numpy()
dayahead63_w_q2 = weights_q2_63.loc[63:1510].to_numpy()
for i in range(0, len(dayahead63_w_q2)):
    dayahead63_port_ret_q2.append(np.multiply(dayahead63_ret_q2[i], daya
head63_w_q2[i]))
dayahead63_port_ret_q2_arr = np.sum(np.array(dayahead63_port_ret_q2), ax
is = 1)
dayahead63_port_ret_q2_arr
```

Out[85]:
```
array([ 0.01740121,  0.0116785 , -0.00496981, ..., -0.00414611,
        0.00514476, -0.00946444])
```

iv) Calculate the standardized outcome, $\tilde{z}_p$, where $\tilde{z}_p = \frac{\tilde{r}_p}{\hat{\sigma}_p}$ where we make the simplifying assumption that $E[\tilde{r}_p] = 0$.

In [86]:
```
standardized_outcomes_63_q2 = dayahead63_port_ret_q2_arr / sd_port_63_ar
r
std_outcomes_63_q2 = pd.DataFrame(standardized_outcomes_63_q2)
std_outcomes_63_q2.index += 63
std_outcomes_63_q2.rename(columns={0: "Standardized Outcome"}, inplace =
True)
std_outcomes_63_q2
```

Out[86]:

	Standardized Outcome
63	1.965812
64	1.285023
65	-0.548099
66	-0.525563
67	0.197667
...	...
1506	0.706105
1507	0.302347
1508	-0.245447
1509	0.304948
1510	-0.561650

1448 rows × 1 columns

b) Compute bias statistics.

In [87]:
```
bias_stat_q2_504 = np.std(standardized_outcomes_504_q2)
bias_stat_q2_252 = np.std(standardized_outcomes_252_q2)
bias_stat_q2_126 = np.std(standardized_outcomes_126_q2)
bias_stat_q2_63 = np.std(standardized_outcomes_63_q2)
print(bias_stat_q2_504, bias_stat_q2_252, bias_stat_q2_126, bias_stat_q2
_63)
```

```
1.1687924023168035 1.0366178517316669 0.9687974333689725 0.946844788289
987
```

Rolling window 63 gives closest bias statistic to 1. Idk why.

Question 3

We now consider the market model approach to estimating volatility. Each step below shouldbe completed using 504, 252, 126, and 63 day rolling windows.

Portfolio from part 2:

In [88]:
```
portfolio_q2_ret
```

Out[88]:

Ticker #	1	2	3	4	5	6	7	8	
0	-0.015048	-0.011842	0.030252	-0.024729	0.000246	-0.004212	-0.072416	-0.038290	0.0
1	0.026042	0.032756	0.008157	0.004754	-0.009089	0.041823	-0.072706	0.033889	0.0
2	0.031472	-0.007478	0.062298	0.014196	0.017848	-0.003608	0.097044	-0.004120	0.0
3	0.012795	-0.007534	0.022848	-0.005184	0.031417	0.022635	0.062683	0.009712	0.0
4	0.045675	0.012042	-0.067014	0.001042	-0.026446	0.034086	0.006064	-0.004809	0.0
...	...	...	...	...	...	...	...	...	
1506	0.000990	0.021250	0.009363	0.011190	0.000242	0.034339	0.019135	0.000760	0.0
1507	0.002308	-0.014688	-0.035250	-0.010060	-0.003867	0.012294	0.016327	0.002658	0.0
1508	-0.002303	-0.004348	-0.007692	0.013720	-0.011160	-0.011861	0.012450	-0.001515	0.0
1509	0.025387	0.016843	0.007752	-0.005013	0.006133	0.012147	0.015470	-0.002275	0.0
1510	-0.000965	-0.011043	0.003846	-0.018136	-0.012924	-0.004290	-0.001953	-0.009122	0.0

1511 rows × 100 columns

In [89]:
```
portfolio_q2_cap
```

Out[89]:

Ticker #	1	2	3	4	5	6	7	
0	13713091.20	32494080.25	1031397.02	237003.60	3004886.52	7.847234e+06	325464.88	6
1	14070202.95	33558466.90	1039809.72	238130.40	2977576.08	8.175431e+06	301801.76	6
2	14513021.52	33307513.95	1104587.51	241510.80	3030720.72	8.145930e+06	331089.72	6
3	14698719.63	33056561.00	1129825.61	240258.80	3125938.20	8.330311e+06	351843.44	6
4	15370089.72	33454624.30	1054111.31	240509.20	3043268.76	8.614257e+06	353977.00	6
...	...	...	...	...	...	...	...	
1506	10467246.96	15921336.52	725246.06	342293.84	3915830.78	1.882777e+08	539539.00	7
1507	10491404.80	15687485.80	699680.80	338850.24	3900689.82	1.905925e+08	548347.80	7
1508	10467246.96	15619279.34	694298.64	343499.10	3857159.56	1.883318e+08	555174.62	7
1509	10732983.20	15882361.40	699680.80	341777.30	3880817.31	1.906195e+08	563763.20	7
1510	10838179.17	15706973.36	702371.88	335578.82	3830662.88	1.898017e+08	562662.10	7

1511 rows × 100 columns

In [90]:
```
df_weights = df_weights_q2_504
df_weights
```

Out[90]:

Ticker #	1	2	3	4	5	6	7	8	9
0	0.018568	0.043998	0.001397	0.000321	0.004069	0.010625	0.000441	0.008199	0.000216
1	0.018809	0.044861	0.001390	0.000318	0.003980	0.010929	0.000403	0.008369	0.000217
2	0.019375	0.044465	0.001475	0.000322	0.004046	0.010875	0.000442	0.008323	0.000218
3	0.019478	0.043805	0.001497	0.000318	0.004142	0.011039	0.000466	0.008342	0.000219
4	0.020277	0.044135	0.001391	0.000317	0.004015	0.011364	0.000467	0.008265	0.000219
...	...	...	...	...	...	...	...	...	...
1505	0.013604	0.020282	0.000935	0.000440	0.005093	0.236809	0.000689	0.009864	0.000120
1506	0.013457	0.020469	0.000932	0.000440	0.005034	0.242061	0.000694	0.009756	0.000119
1507	0.013420	0.020067	0.000895	0.000433	0.004990	0.243796	0.000701	0.009732	0.000123
1508	0.013447	0.020065	0.000892	0.000441	0.004955	0.241941	0.000713	0.009759	0.000125
1509	0.013719	0.020300	0.000894	0.000437	0.004960	0.243645	0.000721	0.009688	0.000126

1510 rows × 100 columns

In [20]:
```
#get risk premium df
eqt_risk_prem_df = portfolio_q2_ret.sub(ffdata['Risk-free rate'], axis=0
)
eqt_risk_prem_df
```

Out[20]:

Ticker #	1	2	3	4	5	6	7	8	
0	-0.015088	-0.011882	0.030212	-0.024769	0.000206	-0.004252	-0.072456	-0.038330	0.0
1	0.026002	0.032716	0.008117	0.004714	-0.009129	0.041783	-0.072746	0.033849	0.0
2	0.031432	-0.007518	0.062258	0.014156	0.017808	-0.003648	0.097004	-0.004160	0.0
3	0.012755	-0.007574	0.022808	-0.005224	0.031377	0.022595	0.062643	0.009672	0.0
4	0.045635	0.012002	-0.067054	0.001002	-0.026486	0.034046	0.006024	-0.004849	0.0
...	...	...	...	...	...	...	...	...	
1506	0.000990	0.021250	0.009363	0.011190	0.000242	0.034339	0.019135	0.000760	0.0
1507	0.002308	-0.014688	-0.035250	-0.010060	-0.003867	0.012294	0.016327	0.002658	0.0
1508	-0.002303	-0.004348	-0.007692	0.013720	-0.011160	-0.011861	0.012450	-0.001515	0.0
1509	0.025387	0.016843	0.007752	-0.005013	0.006133	0.012147	0.015470	-0.002275	0.0
1510	-0.000965	-0.011043	0.003846	-0.018136	-0.012924	-0.004290	-0.001953	-0.009122	0.0

1511 rows × 100 columns

In [21]:
```
mkt_risk_prem_df = pd.DataFrame(ffdata['Market Returns'] - ffdata['Risk-
free rate'])
mkt_risk_prem_df.rename(columns={0:"Market Risk Premium"}, inplace= True
)
mkt_risk_prem_df = mkt_risk_prem_df.reindex(eqt_risk_prem_df.index)
mkt_risk_prem_df
```

Out[21]:

	Market Risk Premium
0	-0.00084
1	0.01216
2	0.00186
3	0.00266
4	0.00476
...	...
1506	0.00520
1507	0.00100
1508	-0.00090
1509	-0.00070
1510	-0.00880

1511 rows × 1 columns

$$\hat{\sigma}_p^2 = Var(\tilde{r}_p) = w^T \hat{\beta} \hat{\sigma}_M^2 \hat{\beta}^T w + w^T \hat{\Delta} w$$

In [129]:
```
def var_portfolio(w, b, m, d):
    return np.dot(np.dot(np.dot(np.dot(np.transpose(w), b), m), np.trans
pose(b)),
                  w) + np.dot(np.dot(np.transpose(w), d), w)
#((((w.T @ beta) @ var) @ beta.T) @ w)
#np.dot(np.dot(np.dot(np.dot(np.transpose(w), b), m), np.transpose(b)),
 w) + np.dot(np.dot(np.transpose(w), d), w)
#(((((np.array(w).T @ np.array(b)) @ np.array(m)) @ np.array(b).T) @ np.
array(w)) +
            #((np.array(w).T @ np.array(d)) @ np.array(w)))
```

Rolling Window 504

i) Use OLS to estimate the market betas for each stock:

In [116]:
```
betas_q3_504 = np.zeros(shape=(1007,1))
for col_index in range(eqt_risk_prem_df.shape[1]):
    ri_minus_rf = eqt_risk_prem_df.iloc[:, col_index]
    rm_minus_rf = mkt_risk_prem_df[["Market Risk Premium"]]
    col_beta = []
    for i in range(1007):
        model = OLS(ri_minus_rf[i:i+503], add_constant(rm_minus_rf[i:i+5
03]))
        res = model.fit()
        beta = res.params[1]
        col_beta.append(beta)
    betas_q3_504 = np.c_[betas_q3_504, col_beta]
betas_q3_504
```

Out[116]:
```
array([[0.        , 1.58613189, 1.40039744, ..., 2.1813825 , 1.6023306
7,
        2.27424324],
       [0.        , 1.58505739, 1.39895521, ..., 2.18106146, 1.6026596
7,
        2.27349407],
       [0.        , 1.56852779, 1.38326599, ..., 2.15678101, 1.6166927
6,
        2.2867706 ],
       ...,
       [0.        , 1.04940178, 1.71999022, ..., 1.42024318, 1.0752302
5,
        1.69069858],
       [0.        , 1.04934783, 1.71996254, ..., 1.42025387, 1.0754745
7,
        1.69115285],
       [0.        , 1.04899455, 1.72005751, ..., 1.42046858, 1.0750120
8,
        1.68990684]])
```

In [117]:
```
betas_df_q3_504 = pd.DataFrame(betas_q3_504).drop(0, axis = 1)
betas_df_q3_504
```

Out[117]:

	1	2	3	4	5	6	7	8	9
0	1.586132	1.400397	2.201703	1.143659	0.984847	1.567119	1.805154	0.771245	1.559725
1	1.585057	1.398955	2.201317	1.139775	0.979738	1.564692	1.801294	0.769244	1.564287
2	1.568528	1.383266	2.199448	1.143749	0.980552	1.562290	1.839123	0.762197	1.543885
3	1.565407	1.383734	2.198504	1.141111	0.979743	1.561133	1.843936	0.760247	1.543489
4	1.563436	1.384451	2.197515	1.141732	0.976663	1.560020	1.838144	0.760024	1.542143
...	...	...	...	...	...	...	...	...	...
1002	1.049304	1.718971	1.326889	1.149744	0.822715	0.948279	1.031868	0.537028	1.998436
1003	1.049207	1.719024	1.326736	1.149722	0.822565	0.948359	1.033234	0.536986	1.998578
1004	1.049402	1.719990	1.325560	1.148348	0.822522	0.949578	1.031047	0.537483	1.995442
1005	1.049348	1.719963	1.325524	1.148276	0.822531	0.949600	1.031096	0.537497	1.995745
1006	1.048995	1.720058	1.325445	1.147740	0.822563	0.949548	1.030476	0.537204	1.996368

1007 rows × 100 columns

ii) Estimate the variance of the market, $\hat{\sigma}_M^2 = Var(\tilde{r}_M)$ and the idiosyncratic variance, $\hat{\sigma}_i^2 = Var(\tilde{\epsilon}_i)$, of each security in your portfolio.

In [108]:
```
var_rm_q3_504 = list(ffdata['Market Returns'].rolling(504).var().dropna
())
var_rm_q3_504 = var_rm_q3_504[:-1]
var_rm_q3_504, len(var_rm_q3_504)
```

Out[108]:
```
([4.601952270503969e-05,
  4.651710625138069e-05,
  4.628661361197901e-05,
  4.628317330146116e-05,
  4.6442464577613765e-05,
  4.643105332291971e-05,
  4.634335636971829e-05,
  4.6310652119031926e-05,
  4.6355279395058304e-05,
  4.6263776704850346e-05,
  4.6293471512196744e-05,
  4.625182618574277e-05,
  4.634404426614923e-05,
  4.6822919625106595e-05,
  4.681742020101622e-05,
  4.683898509735249e-05,
  4.6646033639433334e-05,
  4.660696819085497e-05,
  4.627312123292004e-05,
  4.6274949312064224e-05,
  4.628111233393301e-05,
  4.626996272365815e-05,
  4.6428645271229864e-05,
  4.6272909053614916e-05,
  4.62715080942915e-05,
  4.611079518918246e-05,
  4.6188838043169655e-05,
  4.613817838997769e-05,
  4.594301618858289e-05,
  4.595107766101812e-05,
  4.6048522160686776e-05,
  4.5886758023288935e-05,
  4.594706424153502e-05,
  4.588421329167859e-05,
  4.587302076430316e-05,
  4.5890042443750286e-05,
  4.5908654580453875e-05,
  4.5870437020732834e-05,
  4.587483760137596e-05,
  4.6083900880431776e-05,
  4.6046563791852145e-05,
  4.597632392155015e-05,
  4.5980627307583154e-05,
  4.606971441194114e-05,
  4.612253084666612e-05,
  4.598296301555753e-05,
  4.592953386821933e-05,
  4.554504966234348e-05,
  4.514535891792112e-05,
  4.5031325341601264e-05,
  4.4567332512859425e-05,
  4.453472841522303e-05,
  4.424930318880376e-05,
  4.425007037142234e-05,
  4.416135046861694e-05,
  4.380099304963875e-05,
  4.380509372337412e-05,
```

```
4.378368392817705e-05,
4.326818943481974e-05,
4.333489314115316e-05,
4.316445229417156e-05,
4.312278188014777e-05,
4.3142641295086704e-05,
4.30652485089464e-05,
4.298059579033742e-05,
4.2942963015557526e-05,
4.2917192046135955e-05,
4.309522862823069e-05,
4.308476375871761e-05,
4.321069330051445e-05,
4.2746261439300785e-05,
4.27191476143142e-05,
4.272711631007613e-05,
4.325155034081235e-05,
4.3272495818738444e-05,
4.277414840323149e-05,
4.272482434756549e-05,
4.2352527335984176e-05,
4.23929792672537e-05,
4.237313200164103e-05,
4.2374777367540874e-05,
4.184870759569575e-05,
4.171057334564053e-05,
4.167495499226869e-05,
4.1572814186310786e-05,
4.157667708037498e-05,
4.176690081731839e-05,
4.161276527343881e-05,
4.10742207075011e-05,
4.0677832015841535e-05,
4.090093687872771e-05,
4.125919341096288e-05,
4.1261620120546645e-05,
4.126024787781257e-05,
4.1634207098677855e-05,
4.164616266685609e-05,
4.166708885575444e-05,
4.176136261794321e-05,
4.1762049567673395e-05,
4.174100713181237e-05,
4.158958250497025e-05,
4.164428839660457e-05,
4.208148190223744e-05,
4.223460506800474e-05,
4.252176839755128e-05,
4.251720391934117e-05,
4.3109761076398826e-05,
4.3077869292183485e-05,
4.271520764303078e-05,
4.271987310265399e-05,
4.2504416674555914e-05,
4.294663696393078e-05,
4.305010366373197e-05,
4.297658051689868e-05,
```

Processing math: 100%

```
4.406713938590686e-05,
4.411317203919348e-05,
4.4332219855470426e-05,
4.4303021237653515e-05,
4.4533810312726886e-05,
4.4445517963646765e-05,
4.443594859414946e-05,
4.447857174413842e-05,
4.467049638675889e-05,
4.4699756658462e-05,
4.571135847612744e-05,
4.550118806999282e-05,
4.559987972955923e-05,
4.558893622392636e-05,
4.558958246552439e-05,
4.5533430488497666e-05,
4.5517028898040405e-05,
4.554365327874033e-05,
4.580279462116199e-05,
4.621428744990382e-05,
4.6255239357505824e-05,
4.6239027698886115e-05,
4.623995704345364e-05,
4.688809997159905e-05,
4.671538731894355e-05,
4.689080232888391e-05,
4.725713524409108e-05,
4.7304602227902495e-05,
4.7108522042349145e-05,
4.718068253179344e-05,
4.741613730316521e-05,
4.741617623623347e-05,
4.746195903941438e-05,
4.74384448862382e-05,
4.743100914355147e-05,
4.69029417147907e-05,
4.645670406134629e-05,
4.6497091735302553e-05,
4.6220146265265634e-05,
4.621346109848852e-05,
4.5974075191706984e-05,
4.596965725488348e-05,
4.6013853663731976e-05,
4.615506390230056e-05,
4.585133764871099e-05,
4.5836971464861706e-05,
4.574617150272974e-05,
4.571708854018753e-05,
4.578140486446409e-05,
4.566991160181774e-05,
4.567023746410434e-05,
4.570779288554396e-05,
4.5572848583104626e-05,
4.553193943481973e-05,
4.5514075546719764e-05,
4.5351036006185196e-05,
4.531563472340568e-05,
```

```
4.5506753763135554e-05,
4.552841033166091e-05,
4.552016089968135e-05,
4.5487851778219646e-05,
4.5707225180662136e-05,
4.574361162390743e-05,
4.5667960767143254e-05,
4.564485274858792e-05,
4.56293789248636e-05,
4.560813022657712e-05,
4.555232600429179e-05,
4.524644793934812e-05,
4.52546307867084e-05,
4.5358869008173264e-05,
4.534551796364677e-05,
4.5276029497617556e-05,
4.5242549425668296e-05,
4.5266796009656434e-05,
4.4897740185869e-05,
4.487836974186635e-05,
4.515581617438236e-05,
4.5111755143740814e-05,
4.491235156521199e-05,
4.481791899397275e-05,
4.482032235160474e-05,
4.483973807945983e-05,
4.4917296064880644e-05,
4.47832670248352e-05,
4.477530231310558e-05,
4.47984990059643e-05,
4.463288455773305e-05,
4.463535891792113e-05,
4.460632676165239e-05,
4.446224304963876e-05,
4.446140237937463e-05,
4.417645555239997e-05,
4.3955609044147894e-05,
4.4109569211713924e-05,
4.4102696164284225e-05,
4.410305023036394e-05,
4.428582773202066e-05,
4.403650284799151e-05,
4.36568317081638e-05,
4.387162923254134e-05,
4.38692372747642e-05,
4.388520113446315e-05,
4.393810454731934e-05,
4.379749818549031e-05,
4.366122372905432e-05,
4.374177790400462e-05,
4.368272046293674e-05,
4.361996023856866e-05,
4.362007948341703e-05,
4.340670690144852e-05,
4.335412083057219e-05,
4.336552541891515e-05,
4.333290968474872e-05,
```

```
4.379078607718774e-05,
4.378925112026262e-05,
4.392807093944277e-05,
4.3559976924169335e-05,
4.3573831140143333e-05,
4.3750361955252676e-05,
4.376724084067033e-05,
4.348330761462975e-05,
4.3488416011865386e-05,
4.3461402655495666e-05,
4.346439833222892e-05,
4.334846030957121e-05,
4.331964778787598e-05,
4.339918169554111e-05,
4.3373383824039954e-05,
4.338975038656954e-05,
4.33902980529522e-05,
4.324720782448175e-05,
4.3265539106630126e-05,
4.331449383066686e-05,
4.330604275142801e-05,
4.3294146194263065e-05,
4.329856823345647e-05,
4.334831897503872e-05,
4.335160935182562e-05,
4.313597399728619e-05,
4.290677044873623e-05,
4.282921455394624e-05,
4.2811316229606554e-05,
4.280212802549788e-05,
4.286466471015186e-05,
4.283421940578757e-05,
4.280687553251924e-05,
4.270421029379287e-05,
4.269431411530822e-05,
4.257029237274771e-05,
4.246673609138825e-05,
4.2385926307236016e-05,
4.2464581045473257e-05,
4.266518220044817e-05,
4.265605588690089e-05,
4.258711587617161e-05,
4.2638952909527036e-05,
4.268178165136174e-05,
4.258545615197709e-05,
4.2504735870491405e-05,
4.2484424248161885e-05,
4.2467105935813756e-05,
4.22607352314684e-05,
4.225676543911143e-05,
4.2366604657767744e-05,
4.222812206128316e-05,
4.2308502279971035e-05,
4.2291747570134816e-05,
4.229348338540194e-05,
4.228274779892083e-05,
4.230039094796309e-05,
```

```
4.218759048881322e-05,
4.218937229795835e-05,
4.1808838043169626e-05,
4.177765265549567e-05,
4.4064161065353976e-05,
4.392003088611197e-05,
4.386516456814674e-05,
4.415096405692834e-05,
4.4459760800277773e-05,
4.4983617935245744e-05,
4.482055662059397e-05,
4.4902492189718904e-05,
4.4859653468080474e-05,
4.46785229890499e-05,
4.5520178571428645e-05,
4.551086228659794e-05,
4.549172394206198e-05,
4.543601991227247e-05,
4.552826150241416e-05,
4.559866452081171e-05,
4.61157497869924e-05,
4.606937801760871e-05,
4.590387480671534e-05,
4.588471441194113e-05,
4.5963635330871985e-05,
4.607774432768476e-05,
4.594784810975424 5e-05,
4.563359837009695e-05,
4.564423175234315e-05,
4.571003120167888e-05,
4.570656947205662e-05,
4.570410852346247e-05,
4.569536680709401e-05,
4.565831092808237e-05,
4.5584850736848816e-05,
4.5648159574300294e-05,
4.562707938874695e-05,
4.551817756161451e-05,
4.5247752492978704e-05,
4.471364917637042e-05,
4.472667976269375e-05,
4.4761885472877096e-05,
4.442639839534231e-05,
4.383620274385441e-05,
4.3872136545804614e-05,
4.373888652213709e-05,
4.357486758023296e-05,
4.3770479819495804e-05,
4.347668177443284e-05,
4.341221748871856e-05,
4.338873990185877e-05,
4.339303287418355e-05,
4.309490911672828e-05,
4.3096498469500535e-05,
4.311338630912943e-05,
4.345760670106354e-05,
4.3430784183786255e-05,
```

```
4.344031746031754e-05,
4.320328027864566e-05,
4.322678780491661e-05,
4.306605446684977e-05,
4.306605446684977e-05,
4.2847831581937023e-05,
4.281106594559634e-05,
4.280821732304593e-05,
4.306113789485318e-05,
4.312645839250221e-05,
4.309079518918245e-05,
4.3123779189939804e-05,
4.311656059673712e-05,
4.310749581873844e-05,
4.297104231752351e-05,
4.30370077945029e-05,
4.315398024551114e-05,
4.385016468648435e-05,
4.405483101391658e-05,
4.4039042530531176e-05,
4.4246300254031446e-05,
4.460363718482762e-05,
4.4639229346145425e-05,
4.4716651480008913e-05,
4.4710668686295e-05,
4.467223374041473e-05,
4.5002015052541963e-05,
4.504382273817421e-05,
4.528706033639441e-05,
4.5344742615734244e-05,
4.51715790574017e-05,
4.521994284294242e-05,
4.521816620120554e-05,
4.505481712897227e-05,
4.528330015936137e-05,
4.521639101896565e-05,
4.519743191643796e-05,
4.510140265549568e-05,
4.501488966991708e-05,
4.540967429549693e-05,
4.5192414875824494e-05,
4.5669591340843906e-05,
4.5674173964151674e-05,
4.5699455962636955e-05,
4.57005034870145e-05,
4.570914512922473e-05,
4.5688423743254834e-05,
4.587994745810857e-05,
4.584077187667653e-05,
4.658641089968135e-05,
4.654115304206514e-05,
4.767738126794794e-05,
4.813352989207619e-05,
4.825552932405574e-05,
4.838934377859833e-05,
4.832634001546285e-05,
4.838135125753424e-05,
```

```
4.9671519454700445e-05,
5.0289629682224195e-05,
5.024006043106448e-05,
5.053319685064226e-05,
5.2004094441288854e-05,
5.1956347589068844e-05,
5.1957471677869374e-05,
5.262016330587909e-05,
5.307873895515802e-05,
5.3072180094039016e-05,
5.393311874783055e-05,
5.39336248777179e-05,
5.392701785319835e-05,
5.423271225819693e-05,
5.423300384202728e-05,
5.4491169767111695e-05,
5.459994883871383e-05,
5.57301377449589e-05,
5.653045331187486e-05,
5.650385102085905e-05,
5.676302186878736e-05,
5.6732681963773e-05,
5.695068446464081e-05,
5.694967465050972e-05,
5.7247077179778555e-05,
5.725678358420912e-05,
5.7569962092524304e-05,
5.746113284578253e-05,
5.755341092334888e-05,
5.7442353340275885e-05,
5.7492760224368155e-05,
5.913571318123016e-05,
5.910007759001555e-05,
5.911912509072557e-05,
5.903258859541174e-05,
5.888742209441771e-05,
5.888435352172687e-05,
5.893002982107365e-05,
5.8968528787591996e-05,
5.898902190034406e-05,
5.933382936507946e-05,
5.920423869481533e-05,
5.926197797342936e-05,
5.927146770172624e-05,
5.9320039603648046e-05,
5.874730943702871e-05,
5.876199233172403e-05,
5.873020290952705e-05,
5.8669860677206686e-05,
5.8691862002587735e-05,
5.8706201363249164e-05,
5.880979014800097e-05,
5.866190819369506e-05,
5.864636735144696e-05,
5.96444529253054e-05,
5.9293317554987625e-05,
5.900881425731335e-05,
```

```
5.9021796680236134e-05,
5.848962475149114e-05,
5.882340618984515e-05,
5.881628226671724e-05,
5.8605506603237806e-05,
5.8508869284294323e-05,
5.975251475275341e-05,
5.974089427719407e-05,
5.958217678058641e-05,
5.985555224210303e-05,
6.135609896178493e-05,
6.135202104831338e-05,
6.180297354760336e-05,
6.221751487109101e-05,
6.363265407554682e-05,
6.369486758023299e-05,
6.411581495156057e-05,
6.409895101612557e-05,
6.482903578528837e-05,
6.46457076193633e-05,
6.511180595001431e-05,
6.554547571712593e-05,
6.635565271071992e-05,
6.662402817223653e-05,
6.829490292372759e-05,
6.808219819969085e-05,
6.81690826469754e-05,
6.819855139007237e-05,
6.798842622834433e-05,
6.837923313294844e-05,
6.882698081353161e-05,
6.882537879863685e-05,
6.888730470352498e-05,
7.02219208163717e-05,
7.026620747735816e-05,
7.026736138723226e-05,
7.061657858405135e-05,
7.113275454416369e-05,
7.118313259332902e-05,
7.116823613872331e-05,
7.112532041875741e-05,
7.168082489191845e-05,
7.180960506800479e-05,
7.17769960790812e-05,
7.216541181482546e-05,
7.196032373221003e-05,
7.20457985026351e-05,
7.232978446779653e-05,
7.228647350026835e-05,
7.312547283757782e-05,
7.307331120420346e-05,
7.367128782858416e-05,
7.368275201962838e-05,
7.378389184732884e-05,
7.418011608917931e-05,
7.437355198176032e-05,
7.552066584619278e-05,
```

```
7.560670165514858e-05,
7.713398261226304e-05,
7.71625478872796e-05,
7.722728482280928e-05,
7.748675802328895e-05,
7.77716665088833e-05,
7.809100823629661e-05,
7.865532767679646e-05,
7.860760184922223e-05,
7.857461417999947e-05,
7.900623114487688e-05,
7.949560600681635e-05,
7.96615960980152e-05,
8.142002840102254e-05,
8.15014597731075e-05,
8.161004244375031e-05,
8.140137362333944e-05,
8.137965287639255e-05,
8.144608251285947e-05,
8.185116629587564e-05,
8.215690286850338e-05,
8.198945434535655e-05,
8.196908438259343e-05,
8.201130546088559e-05,
8.229984521442783e-05,
8.237085388462884e-05,
8.273714565779934e-05,
8.284493972671916e-05,
8.283835238568599e-05,
8.294213575688736e-05,
8.412327365174046e-05,
8.395760543879591e-05,
8.400327144277204e-05,
8.409004824229239e-05,
8.494807015052554e-05,
8.504207859982972e-05,
8.571410394774223e-05,
8.820722518066217e-05,
8.820661033797228e-05,
8.829856097541745e-05,
8.889948274637898e-05,
8.929740789390653e-05,
9.243053358420915e-05,
9.378012354444772e-05,
9.467894438922035e-05,
9.519065602417255e-05,
9.518616266685613e-05,
9.526716131780758e-05,
9.546797212755228e-05,
9.559574536905564e-05,
9.5591070718546e-05,
9.748367757739292e-05,
9.748228517782208e-05,
9.747853198270706e-05,
9.748457220959968e-05,
9.743288266433161e-05,
9.741143993183768e-05,
```

```
9.757574233172407e-05,
9.742753475180676e-05,
9.820337968222425e-05,
9.807557981476236e-05,
9.812010019249596e-05,
9.926570572596183e-05,
9.926262283442218e-05,
9.922254473161049e-05,
9.919191706901463e-05,
9.936833854018758e-05,
9.937002918993986e-05,
9.939989712518554e-05,
9.945493037804931e-05,
9.944784562466485e-05,
9.951034045725661e-05,
9.951471642368029e-05,
9.990635046861702e-05,
9.985882952286298e-05,
9.985252469311133e-05,
9.996178358420918e-05,
0.00010026669696109075,
0.0001003051860266971 1,
0.00010034990485657499,
0.00010057585913092885,
0.00010019701221243979,
9.986957998043499e-05,
0.00010009822300325049,
0.00010010322493609784,
9.949413739783536e-05,
9.945556549591355e-05,
9.991776669349e-05,
9.984192117138454e-05,
0.00010009328915396525,
0.00010013649622108633,
9.98342146722839e-05,
9.979999431979566e-05,
9.933376171542192e-05,
9.934856511723325e-05,
9.9102261628641e-05,
9.90943745463727e-05,
9.915146924011501e-05,
0.00010058134522231703,
0.0001004883625232732,
0.00010053540029663305,
0.0001009844493357318,
0.00010052959244532816,
0.0001005884603095713,
0.0001005627429076337,
9.95033663100762e-05,
9.960436807725092e-05,
9.939093376250447e-05,
0.00010001852405408831,
9.976059247688486e-05,
9.978848985452378e-05,
9.985824986588419e-05,
0.00010120166414213148,
0.00010101733562908278,
```

```
0.00010097632616996448,
9.990130072738187e-05,
0.0001007878934330526,
0.00010068243806999287,
0.00010068243806999287,
0.00010125553784436252,
0.00010189400166461561,
0.00010200565038341392,
0.00010211404130770941,
0.00010200948968885115,
0.00010186482213859711,
0.0001029562690129068,
0.00010310710305626568,
0.00010310374073022196,
0.00010236647969326908,
0.00010236015320773781,
0.00010219745207327474,
0.0001027941717551833,
0.00010274566821294466,
0.0001033008438259334,
0.00010413594038940969,
0.00010434561945785615,
0.00010462341281675039,
0.00010461917159739983,
0.00010489324481681351,
0.00010608527485878391,
0.00010614524282874193,
0.00010669481109375503,
0.00010739742398781922,
0.0001074462989917638,
0.00010761041082867881,
0.00010756520776136839,
0.0001076149471030958,
0.00010719304490517223,
0.00010739036885827899,
0.00010754818501688291,
0.00010762457114456129,
0.00010762183454826604,
0.0001078018952554515,
0.00010844218936381718,
0.00010846407519170698,
0.00010864225216163345,
0.00010898949020164734,
0.00010925257723500276,
0.00010934743396762291,
0.00010936431596926386,
0.00011098318833033553,
0.00011100084571933487,
0.00011116883835873656,
0.0001135502658651268,
0.00011368604369812878,
0.00011392277442487939,
0.00011372608314399328,
0.0001178464394190414,
0.00011830979440815435,
0.00012244184886711488,
0.00012635118227145078,
```

```
0.00013051215812269254,
0.0001331724960948595,
0.00013361506958250508,
0.0001336065592161319,
0.00013403425715547993,
0.00013393751455552413,
0.00014752005865600065,
0.00015191461879516567,
0.0001519567262693680 7,
0.00015603274255262094,
0.00015642711536337545,
0.0001596393745463727,
0.00016620089143709187,
0.0001664279994635364,
0.00017690673857647779,
0.00017714276752185315,
0.00020375273497901493,
0.00020365729302754913,
0.00021950135504433728,
0.00022271539840323156,
0.00022268675183028825,
0.0002270969085487079,
0.00022896617296222678,
0.00023576987724446984,
0.00023570469843636608,
0.00023780068904036115,
0.00024035838204897615,
0.0002586663176496577,
0.00025855834879611237,
0.0002606554640017042,
0.00026136792763261713,
0.0002612213020290 9543,
0.0002643174451702485,
0.00026931245526838984,
0.00027372550411814843,
0.00027516632770835327,
0.00027545590705765427,
0.0002765353791142039,
0.00028212127800656404,
0.0002916006692779831,
0.00029514822394995137,
0.0002961956559058351,
0.0002962811074820918,
0.0003042812665278175,
0.0003139656933005147,
0.00032172563034491496,
0.0003308313263277481,
0.00033104799453280345,
0.0003337637261746981,
0.00033398237558774366,
0.00034956082611474037,
0.0003525320874751493,
0.000353782885188867,
0.00035528767360124984,
0.0003580255353198273,
0.00036143567026412975,
0.0003623738615529 05,
```

```
0.00036285754220707526,
0.0003645717276894193,
0.00036482364669128116,
0.0003651664034444131,
0.000370785791284042,
0.0003708207056076243,
0.0003715621911388812,
0.0003715975333712015,
0.00037228387389946055,
0.0003723548484805453017,
0.0003723796771355993,
0.0003724563078276375,
0.00037234164725930154,
0.0003736564381567738,
0.000374289462865633,
0.00037627527911893735,
0.0003762730038420274,
0.0003764676737984792,
0.00037822946002556116,
0.000378329400383414,
0.0003792582504575722,
0.00038033954542585774,
0.0003802904118148255 5,
0.0003826611814430877 3,
0.000382729991124680 7,
0.00038287785225157015,
0.0003886876906418633 3,
0.000392531627378585 9,
0.00039305794502824347,
0.000392996972293225,
0.00039289465918773095,
0.0003931881851746665,
0.0003955713827747801,
0.000397535642928145 6,
0.0003982494186862951,
0.00039815105332291983,
0.0003986168531667141 6,
0.000398642334524598 6,
0.00039918096559531715,
0.000400832806336583 8,
0.00040084186626274115,
0.0004048849549922687 4,
0.000404998454944933 7,
0.00040487030921613197,
0.0004048691100618512 6,
0.0004088324935703242 4,
0.00040884419135977796,
0.0004090136305579224 5,
0.000409245846784373 3,
0.00041174738043958485,
0.000415234768176654 5,
0.00041541831672662455,
0.0004134737129208875 6,
0.000414079857994888,
0.00041885764965761006,
0.00041864804975701365,
0.00041986900560920214,
```

```
0.0004226467849648145,
0.0004226623722742909 6,
0.000422729339163432 3,
0.0004312676095411658 5,
0.000431289661041686 5,
0.0004343082943608193 6,
0.000434329273249392 7,
0.0004342970197465999,
0.0004363654043595571,
0.0004371601018886681 5,
0.0004371492644135190 3,
0.0004372913499558207 5,
0.000447214974872984 4,
0.000447928274637887 1,
0.0004481204102369908 5,
0.000449464209938780 1,
0.000450247601849222 2,
0.000452729591340843 9,
0.0004531972260090252 6,
0.0004538291958960523 6,
0.000455795466565685 3,
0.000456104011131623,
0.000456224179210451 6,
0.000457377414875824 4,
0.0004577551200732115 3,
0.000460983520740635 6,
0.000460930057117611 8,
0.000461601480008835 9,
0.0004617085693773865 7,
0.000462293157089210 85,
0.000462389036850326 67,
0.000466292094062608 5,
0.000467217602914260 5,
0.0004672620675549875 6,
0.000467458882735334,
0.0004680525304916532 6,
0.000468274572209599 55,
0.0004682821297216699 7,
0.000469415904414781 1,
0.0004694047968932436 5,
0.000469339380226577,
0.0004719855996954778 6,
0.000471976420051121 8,
0.000472512905266811 8,
0.000472945583285998 3,
0.0004744813990659218 7,
0.000474949463593044 905,
0.000474718175234308 4,
0.000476575983740414 6,
0.000476928325246931 1,
0.0004770018412935087 7,
0.000479307750284010 2,
0.000479203487606109 34,
0.0004791768379800561 6,
0.000479690289335416 04,
0.000479691002516646 1,
0.0004811720065322351 3,
```

```
0.00048163636431411525,
0.00048211484805452993,
0.00048243625575909620,
0.00048398432815803590,
0.00048395250177506390,
0.00048448848587049134,
0.00048485906639527910,
0.00048471442452428290,
0.00048410072797342926,
0.00048392489996528770,
0.00048393141811827450,
0.00048383846251853960,
0.00048341489286503200,
0.00048458341506516455,
0.00048477421372558296,
0.00048478509605068000,
0.00048489000599577134,
0.00048464632235160460,
0.00048670570541039474,
0.00048651835068951370,
0.00048667010681940100,
0.00048765081936949730,
0.00048746599009119883,
0.00048760588717693836,
0.00048770686736722520,
0.00048751496670769033,
0.00048900632171258170,
0.00048900200511218400,
0.00048966249928997440,
0.00048966831755498750,
0.00048939646119315840,
0.00048934585487077530,
0.00048995661361197890,
0.00049005788696393060,
0.00049209876723784270,
0.00049230899267095830,
0.00049230758776704850,
0.00049264940038341380,
0.00049241727314683330,
0.00049239547271924000,
0.00049288492154217540,
0.00049291510350594830,
0.00049184918047271920,
0.00049145630218687860,
0.00049134339293603450,
0.00049153411515036750,
0.00049151498118432260,
0.00049211362393101700,
0.00049081844914639140,
0.00049015842054813970,
0.00049016815105399330,
0.00049006121836441660,
0.00048867756887247940,
0.00048903915364953130,
0.00048936512354444750,
0.00048880217551831850,
0.00048850628416011850,
```

```
0.0004898159896178483,
0.0004889236150951433,
0.0004890261421550063,
0.0004892637343794376,
0.0004896194622345292,
0.0004896159536432199,
0.0004893215084098582,
0.0004891907664725929,
0.0004881392786140299,
0.0004871847880573699,
0.00048731359837009665,
0.0004879465150367633 5,
0.0004877010129303541 6,
0.000487755254149704 7,
0.0004881300719098107,
0.000487851174816182,
0.0004879723246236862,
0.00048786178484647646,
0.00048786178484647646,
0.0004878417740777554 3,
0.000487922587648710 6,
0.000488406540716021,
0.0004867087067673323,
0.00048664923786645187,
0.0004866280383966043,
0.00048673307831581915,
0.000486897877457477 2,
0.000487199933573164,
0.0004872095313831296,
0.0004878346812774147,
0.000487823083719902 6,
0.00048744724084067006,
0.0004888803502398306,
0.0004888822922780773,
0.0004894084691454446,
0.0004895651292246518,
0.0004895745750497015,
0.0004896405563050266,
0.0004897077004244371,
0.0004896966028827035,
0.000489639499786992,
0.0004901657246599762,
0.0004900881780744103,
0.0004901926024409097,
0.0004904046435277222,
0.0004893399190570856,
0.0004894501390072261,
0.0004894585061062194,
0.0004897488874688374,
0.0004900807554277506,
0.0004897559897361857,
0.0004907855088122058,
0.0004918740158651263,
0.0004930413902300479,
0.0004917926033876105,
0.0004918587648710906,
0.0004917798958234715,
```

```
 0.0004922337038088924,
 0.0004908193192827161,
 0.0004919358076540752,
 0.0004914925645728798,
 0.0004911698961784844,
 0.0004898662709063078,
 0.0004899007156663192,
 0.0004900058619710306,
 0.0004899663176496573,
 0.0004892716002398305,
 0.0004896243935987248,
 0.0004891876909968756,
 0.000488999491937265,
 0.0004881726459891 4416,
 0.0004879394849553 4694,
 0.0004868720587191135,
 0.0004869063371753602,
 0.0004872083459560 0965,
 0.0004871616799204768,
 0.0004872089374467 4777,
 0.0004868565577960803,
 0.0004863593861829 0224,
 0.0004865707354286973,
 0.0004864736775773 9243,
 0.0004852628883445356,
 0.0004852208343589255,
 0.0004853718872479406,
 0.0004850595313436 8355,
 0.0004845809135267755,
 0.0004847957975559339,
 0.0004848772165420174,
 ...],
1007)
```

Processing math: 100%

In [50]:
```
var_e_q3_504 = np.zeros(shape=(1007,1))
for col_index in range(eqt_risk_prem_df.shape[1]):
    ri_minus_rf = eqt_risk_prem_df.iloc[:, col_index]
    rm_minus_rf = mkt_risk_prem_df[["Market Risk Premium"]]
    col_vars = []
    for i in range(1007):
        model = OLS(ri_minus_rf[i:i+503], add_constant(rm_minus_rf[i:i+5
03]))
        res = model.fit()
        varis = res.resid
        col_vars.append(np.var(varis))
    var_e_q3_504 = np.c_[var_e_q3_504, col_vars]
var_e_q3_504
```

Out[50]:
```
array([[0.        , 0.00032241, 0.00016527, ..., 0.00090192, 0.0007921
4,
        0.00088757],
       [0.        , 0.00032204, 0.00016512, ..., 0.00090189, 0.0007920
7,
        0.00088745],
       [0.        , 0.00032268, 0.0001648 , ..., 0.00090647, 0.0007862
8,
        0.00088911],
       ...,
       [0.        , 0.00027802, 0.00092841, ..., 0.00367989, 0.0011591
9,
        0.00199991],
       [0.        , 0.00027791, 0.00092886, ..., 0.00368292, 0.0011573
3,
        0.00199361],
       [0.        , 0.00027757, 0.00092886, ..., 0.0036828 , 0.0011573
5,
        0.00198985]])
```

In [52]:
```
vars_df_q3_504 = pd.DataFrame(var_e_q3_504).drop(0, axis = 1)
vars_df_q3_504
```

Out[52]:

	1	2	3	4	5	6	7	8	9
0	0.000322	0.000165	0.000687	0.000422	0.000270	0.000510	0.000968	0.000170	0.000248
1	0.000322	0.000165	0.000685	0.000421	0.000271	0.000510	0.000958	0.000167	0.000249
2	0.000323	0.000165	0.000685	0.000421	0.000271	0.000510	0.000940	0.000166	0.000251
3	0.000321	0.000165	0.000679	0.000421	0.000271	0.000510	0.000929	0.000166	0.000251
4	0.000322	0.000165	0.000679	0.000421	0.000269	0.000510	0.000922	0.000166	0.000251
...	...	...	...	...	...	...	...	...	...
1002	0.000278	0.000929	0.003222	0.000736	0.000539	0.000473	0.001003	0.000275	0.002759
1003	0.000278	0.000928	0.003221	0.000733	0.000539	0.000473	0.001009	0.000275	0.002759
1004	0.000278	0.000928	0.003220	0.000731	0.000539	0.000475	0.001006	0.000275	0.002754
1005	0.000278	0.000929	0.003222	0.000732	0.000539	0.000475	0.001006	0.000275	0.002755
1006	0.000278	0.000929	0.003222	0.000731	0.000539	0.000475	0.001006	0.000275	0.002754

1007 rows × 100 columns

iii) Using the market capitalization weights (from the last day in the rolling window)of your securities, estimate the variance and standard deviation of your portfolio.

Formula:

$$\hat{\sigma}_p^2 = Var(\tilde{r}_p) = w^T \hat{\beta} \hat{\sigma}_M^2 \hat{\beta}^T w + w^T \hat{\Delta} w$$

In [99]:
```
df_weights_504 = df_weights.tail(1007)
df_weights_504
```

Out[99]:

Ticker #	1	2	3	4	5	6	7	8	9
503	0.018565	0.027973	0.001530	0.000238	0.005111	0.066085	0.000360	0.009388	0.000279
504	0.018351	0.027784	0.001531	0.000238	0.005027	0.067496	0.000358	0.009413	0.000273
505	0.018299	0.027789	0.001563	0.000236	0.005035	0.067324	0.000378	0.009300	0.000273
506	0.018792	0.028031	0.001533	0.000237	0.005063	0.066842	0.000377	0.009191	0.000277
507	0.018690	0.027641	0.001549	0.000233	0.005003	0.067839	0.000374	0.008993	0.000275
...	...	...	...	...	...	...	...	...	...
1505	0.013604	0.020282	0.000935	0.000440	0.005093	0.236809	0.000689	0.009864	0.000120
1506	0.013457	0.020469	0.000932	0.000440	0.005034	0.242061	0.000694	0.009756	0.000119
1507	0.013420	0.020067	0.000895	0.000433	0.004990	0.243796	0.000701	0.009732	0.000123
1508	0.013447	0.020065	0.000892	0.000441	0.004955	0.241941	0.000713	0.009759	0.000125
1509	0.013719	0.020300	0.000894	0.000437	0.004960	0.243645	0.000721	0.009688	0.000126

1007 rows × 100 columns

In [133]:
```
var_port_q3_504 = []
for i in range(len(df_weights_504)):
    diag_mat = np.diag(vars_df_q3_504.iloc[i])
    res = var_portfolio(df_weights_504.iloc[i], betas_df_q3_504.iloc[i],
var_rm_q3_504[i], diag_mat)
    var_port_q3_504.append(res)
arr_var_q3_504 = np.array(var_port_q3_504)
arr_var_q3_504
```

Out[133]:
```
array([6.63549209e-05, 6.72981929e-05, 6.70513956e-05, ...,
       4.47347672e-04, 4.46184599e-04, 4.46173349e-04])
```

In [134]:
```
arr_sd_q3_504 = np.sqrt(arr_var_q3_504)
arr_sd_q3_504
```

Out[134]:
```
array([0.00814585, 0.00820355, 0.00818849, ..., 0.0211506 , 0.02112308,
       0.02112282])
```

iv) Using the market capitalization weights and returns (from the day following the last day in the rolling window) of your securities, calculate the one-day ahead return of the portfolio, $\bar{r}_p$.

In [131]:
```
#getting one-day ahead returns array
dayahead504_port_ret_q3 = []
dayahead504_ret_q3 = portfolio_q2_ret.loc[504:1510].to_numpy()
dayahead504_w_q3 = weights_q2_504.loc[504:1510].to_numpy()
for i in range(0, len(dayahead504_w_q2)):
    dayahead504_port_ret_q3.append(np.multiply(dayahead504_ret_q3[i], da
yahead504_w_q3[i]))
dayahead504_port_ret_q3_arr = np.sum(np.array(dayahead504_port_ret_q3),
axis = 1)
dayahead504_port_ret_q3_arr
```

Out[131]:
```
array([ 0.01826034,  0.00564215, -0.00055066, ..., -0.00414611,
        0.00514476, -0.00946444])
```

v) Calculate the standardized outcome, z_p, where $z_p = \frac{\tilde{r}_p}{\hat{\sigma}_p}$ where we make the simplifying assumption that $E[\tilde{r}_p] = 0$.

In [147]:
```
standardized_outcomes_504_q3 = dayahead504_port_ret_q3_arr / arr_sd_q3_5
04
std_outcomes_504_q3 = pd.DataFrame(standardized_outcomes_504_q3)
std_outcomes_504_q3.index += 504
std_outcomes_504_q3.rename(columns={0: "Standardized Outcome"}, inplace
= True)
std_outcomes_504_q3
```

Out[147]:

	Standardized Outcome
504	2.241673
505	0.687769
506	-0.067248
507	1.343726
508	0.453598
...	...
1506	0.572534
1507	0.245258
1508	-0.196028
1509	0.243561
1510	-0.448067

1007 rows × 1 columns

Rolling Window 252

i) Use OLS to estimate the market betas for each stock:

In [137]:
```
betas_q3_252 = np.zeros(shape=(1259,1))
for col_index in range(eqt_risk_prem_df.shape[1]):
    ri_minus_rf = eqt_risk_prem_df.iloc[:, col_index]
    rm_minus_rf = mkt_risk_prem_df["Market Risk Premium"]
    col_beta = []
    for i in range(1259):
        model = OLS(ri_minus_rf[i:i+251], add_constant(rm_minus_rf[i:i+2
51]))
        res = model.fit()
        beta = res.params[1]
        col_beta.append(beta)
    betas_q3_252 = np.c_[betas_q3_252, col_beta]
betas_q3_252
```

Out[137]:
```
array([[0.        , 1.9319157 , 1.56218804, ..., 2.6725418 , 1.8120748
6,
        2.30861338],
       [0.        , 1.93020096, 1.56099466, ..., 2.67307771, 1.8153182
3,
        2.3086432 ],
       [0.        , 1.91678797, 1.5441574 , ..., 2.66836787, 1.8508778
7,
        2.30164876],
       ...,
       [0.        , 1.13115708, 1.93123882, ..., 1.57264432, 1.1709713
4,
        1.68377417],
       [0.        , 1.12920755, 1.93157658, ..., 1.57335547, 1.1702091
4,
        1.67916135],
       [0.        , 1.14018041, 1.91684191, ..., 1.58306463, 1.1652043
9,
        1.69638467]])
```

In [138]:
```
betas_df_q3_252 = pd.DataFrame(betas_q3_252).drop(0, axis = 1)
betas_df_q3_252
```

Out[138]:

	1	2	3	4	5	6	7	8	9
0	1.931916	1.562188	2.317098	1.147359	0.955836	1.385805	2.060276	0.899883	1.342231
1	1.930201	1.560995	2.321713	1.143028	0.955016	1.386188	2.056838	0.895685	1.346941
2	1.916788	1.544157	2.322160	1.195700	0.968974	1.372672	2.155587	0.874056	1.352562
3	1.913509	1.542065	2.354886	1.184452	0.960986	1.348913	2.160270	0.871213	1.352927
4	1.905218	1.541584	2.366488	1.183212	0.951547	1.339436	2.155961	0.868018	1.363342
...	...	...	...	...	...	...	...	...	...
1254	1.129996	1.931494	1.456411	1.198005	0.573958	0.833084	0.977473	0.467603	2.888859
1255	1.130210	1.932652	1.455659	1.196772	0.573242	0.833962	0.980778	0.467705	2.887693
1256	1.131157	1.931239	1.455531	1.197119	0.573145	0.835204	0.977920	0.467129	2.889860
1257	1.129208	1.931577	1.453296	1.198668	0.573467	0.835939	0.976104	0.466884	2.887394
1258	1.140180	1.916842	1.456712	1.199883	0.564602	0.844310	0.976781	0.461114	2.883868

1259 rows × 100 columns

ii) Estimate the variance of the market, $\hat{\sigma}_M^2 = Var(\tilde{r}_M)$ and the idiosyncratic variance, $\hat{\sigma}_i^2 = Var(\tilde{\epsilon}_i)$, of each security in your portfolio.

In [139]:
```
var_rm_q3_252 = list(ffdata['Market Returns'].rolling(252).var().dropna
())
var_rm_q3_252 = var_rm_q3_252[:-1]
var_rm_q3_252, len(var_rm_q3_252)
```

Out[139]: ([5.00546569278442e-05,
5.049132406880417e-05,
5.063620502118512e-05,
5.079488316574972e-05,
5.0810256118383625e-05,
5.075544472902045e-05,
5.064044820717134e-05,
5.072227834692976e-05,
5.068556757098592e-05,
5.069892035034467e-05,
5.0837047366091214e-05,
5.097263264402708e-05,
5.1291223676721704e-05,
5.1411035698475954e-05,
5.146795089483339e-05,
5.1564262790109426e-05,
5.116043255549233e-05,
5.100625482198193e-05,
5.0179867672168484e-05,
5.021428998292546e-05,
5.054503256814016e-05,
5.0687419686334043e-05,
5.073656232214003e-05,
5.034946104471006e-05,
5.076353569847596e-05,
4.999692831847216e-05,
4.9997599443495875e-05,
5.0225226554101076e-05,
4.989830946057043e-05,
5.002678935053439e-05,
4.9900513501549374e-05,
4.956247438816166e-05,
4.9473345190665926e-05,
4.9567699203187273e-05,
4.950778157212422e-05,
5.0181766110162545e-05,
5.025725083791819e-05,
5.037231818756721e-05,
5.070487130841714e-05,
5.0852305065452494e-05,
5.056049120976413e-05,
5.0418140137861274e-05,
5.041487699993678e-05,
5.0681951558844e-05,
5.06709802061595e-05,
5.048133877189655e-05,
5.067688658066149e-05,
4.971740893568584e-05,
4.901298741541771e-05,
4.851721542401823e-05,
4.7682187282615585e-05,
4.7878267722759765e-05,
4.731410674761274e-05,
4.731559650287739e-05,
4.702358992601026e-05,
4.652527161196485e-05,
4.655804211724532e-05,

```
4.6501739549737565e-05,
4.544870596977171e-05,
4.573570796180359e-05,
4.5725525042686405e-05,
4.5635080471763746e-05,
4.5771085024979454e-05,
4.5621620660216285e-05,
4.538668737747424e-05,
4.524916761525328e-05,
4.529459874786569e-05,
4.5515594763801943e-05,
4.55011552203883e-05,
4.5534608708025054e-05,
4.519719724277494e-05,
4.566335736419402e-05,
4.6689862771137675e-05,
4.6633369221526595e-05,
4.686323452222854e-05,
4.654545674445077e-05,
4.767445819895023e-05,
4.707013659647126e-05,
4.73688805097072e-05,
4.7635645987478645e-05,
4.764220830961866e-05,
4.7175793966989175e-05,
4.723832985518244e-05,
4.715507288307088e-05,
4.6890956017201025e-05,
4.748665196357426e-05,
4.7479322551065566e-05,
4.7180942736988543e-05,
4.626101309049515e-05,
4.588098321001706e-05,
4.55490246948713e-05,
4.6031979225953315e-05,
4.614691646113956e-05,
4.647841902232339e-05,
4.613538164801112e-05,
4.639012379055207e-05,
4.645540552077404e-05,
4.647043382027445e-05,
4.647362976664769e-05,
4.643116470625434e-05,
4.5513226459242386e-05,
4.565691314108644e-05,
4.556736656548408e-05,
4.5656823499652175e-05,
4.594480933409219e-05,
4.5931154746094964e-05,
4.576739692025546e-05,
4.564673290963129e-05,
4.472921061784605e-05,
4.475745952697145e-05,
4.4377406406121516e-05,
4.435224182634538e-05,
4.385032868525893e-05,
4.365180168216022e-05,
```

```
4.367201843419968e-05,
4.375430278884459e-05,
4.379178144564595e-05,
4.3731630936571135e-05,
4.370806488332381e-05,
4.342560978308982 5e-05,
4.382920176437105e-05,
4.4003575539113355e-05,
4.39791873774742e-05,
4.429984996521846e-05,
4.41996977170682e-05,
4.3934649971542364e-05,
4.3957231866186014e-05,
4.38629309745146e-05,
4.402742300638712e-05,
4.367126209447919e-05,
4.4154587206728606e-05,
4.432838297603234e-05,
4.434438247011949e-05,
4.430604439385312e-05,
4.424819578827543e-05,
4.4139545943211244e-05,
4.4252185385442324e-05,
4.4280552077404635e-05,
4.347248276734329e-05,
4.3696122652248116e-05,
4.332962420160624e-05,
4.3313231360273154e-05,
4.294594890280146e-05,
4.2997250837918126e-05,
4.301691946499712e-05,
4.321047666476946e-05,
4.3183648105988707e-05,
4.313748229304998e-05,
4.310638762410671e-05,
4.227162002782517e-05,
4.1634109119079205e-05,
4.1693465503067064e-05,
4.11642661101625e-05,
4.1103594985138784e-05,
4.066166745715547e-05,
4.0797029026750116e-05,
4.011629482071711e-05,
4.069451131980014e-05,
4.010814962372729e-05,
4.008206033010812e-05,
3.986187108708023e-05,
3.98095516347309e-05,
3.986024331246441e-05,
3.974524552583315e-05,
3.9751889584519044e-05,
3.9941420824637936e-05,
3.97591277746158e-05,
3.9715765509390986e-05,
4.017145070511602e-05,
3.978625102763547e-05,
3.976468775690885e-05,
```

```
4.007408081957881e-05,
3.9987233763359244e-05,
4.0026383829760306e-05,
4.015496284702458e-05,
4.01358426611016e-05,
4.036489644596217e-05,
4.023903813318154e-05,
4.0189146113956855e-05,
4.0331673148675124e-05,
4.0323544393853144e-05,
4.0385474609498506e-05,
4.0059207455890707e-05,
4.0005878707392636e-05,
4.0005103870233345e-05,
3.98060323784228e-05,
3.9665822266489585e-05,
3.9595531998988166e-05,
3.9849402390438235e-05,
3.905653323215075e-05,
3.9017089103901845e-05,
3.942091491178144e-05,
4.043624913046227e-05,
4.0166994245241246e-05,
4.0041420824637946e-05,
4.0279580566622383e-05,
4.0288918453171425e-05,
4.0347559602858395e-05,
4.008282805286788e-05,
4.033342487194079e-05,
4.030288496806423e-05,
4.048998087017009e-05,
4.122114763169542e-05,
4.2084570290267476e-05,
4.175048551824446e-05,
4.2832279769809634e-05,
4.223053105040155e-05,
4.174394280022764e-05,
4.237271659394167e-05,
4.320729826724844e-05,
4.3534531714412165e-05,
4.356371260987792e-05,
4.359099158919874e-05,
4.2880277461582216e-05,
4.2833689369506074e-05,
4.283818804148483e-05,
4.289516647694932e-05,
4.2904154176942996e-05,
4.2782735091380485e-05,
4.252925615000314e-05,
4.253541816859543e-05,
4.2476351735913464e-05,
4.2349737557705656e-05,
4.2767631695440435e-05,
4.23960024979447e-05,
4.240535176753301 5e-05,
4.250242774932015e-05,
4.244486482640862e-05,
```

```
4.2452776038702306e-05,
4.285868383608421e-05,
4.280353301081387e-05,
4.21391121229368e-05,
4.277751596787451e-05,
4.2775759659773586e-05,
4.2801549832416344e-05,
4.222899955732623e-05,
4.226924729652815e-05,
4.221236767216845e-05,
4.2229290299120956e-05,
4.197164658825015e-05,
4.196558195788273e-05,
4.194641039018527e-05,
4.1938183772845104e-05,
4.188374438752922e-05,
4.210743881616389e-05,
4.181025611838358e-05,
4.181545374059316e-05,
4.188529105799024e-05,
4.181406358692213e-05,
4.199135442357551e-05,
4.20155410105609e-05,
4.204401109846326e-05,
4.212877758806043e-05,
4.271875213431983e-05,
4.212083270094223e-05,
4.195185670018337e-05,
4.2253505343704515e-05,
4.2278031998988146e-05,
4.222479020426228e-05,
4.206470546385882e-05,
4.220279437804335e-05,
4.19917401821286e-05,
4.192356842471381e-05,
4.171173717827101e-05,
4.156228972996898e-05,
4.239846882312019e-05,
4.232173907544422e-05,
4.225477834692971e-05,
4.226839056472519e-05,
4.235934911149051e-05,
4.252495668121163e-05,
4.248564156074113e-05,
4.2185784323025326e-05,
4.202699803958765e-05,
4.230278884462148e-05,
4.237998276734329e-05,
4.1963372541579675e-05,
4.2408364004300224e-05,
4.256401995193824e-05,
4.222992774932015e-05,
4.216482814772652e-05,
4.2058392461898405e-05,
4.237984490608989e-05,
4.239271912350594e-05,
4.2602441029532626e-05,
```

```
4.237984490608989e-05,
4.241924239549734e-05,
4.1765843767785964e-05,
4.173565594763798e-05,
4.154723328906592e-05,
4.122688847783466e-05,
4.1497842914058016e-05,
4.1712468696641966e-05,
4.17159721431733e-05,
4.172808369695816e-05,
4.1640464175045807e-05,
4.175591285651042e-05,
4.166619237336365e-05,
4.13648732055903e-05,
4.155274647441974e-05,
4.145593040536264e-05,
4.172477644975649e-05,
4.163003399102e-05,
4.136509896920251e-05,
4.135362976664765e-05,
4.135362976664765e-05,
4.147094336937958e-05,
4.1243075159678704e-05,
4.1216761525327227e-05,
4.1231034433693756e-05,
4.1242145544804876e-05,
4.107930357933342e-05,
4.075135015493578e-05,
4.0756423670397736e-05,
4.065185527730344e-05,
4.0642815879339744e-05,
4.0700322993739285e-05,
4.0754693448428464e-05,
4.0675179124770713e-05,
4.0808264086511055e-05,
4.080206159489023e-05,
4.1035019762220915e-05,
4.043192231075693e-05,
3.988761699234802e-05,
3.8829509106431366e-05,
3.9905355087586117e-05,
3.973688658066144e-05,
3.903821981913611e-05,
3.7865567570985854e-05,
3.767755691519632e-05,
3.7500145924239506e-05,
3.7164844747992116e-05,
3.716309745146395e-05,
3.6558430405362634e-05,
3.6276294662619327e-05,
3.6303894264212954e-05,
3.635295706064627e-05,
3.578929282868522e-05,
3.615770236514257e-05,
3.6158383608423416e-05,
3.6028697906785526e-05,
3.5610608360209925e-05,
```

```
3.639796559792573e-05,
3.6625103870233326e-05,
3.650710870802502e-05,
3.619157085941944e-05,
3.729887102384112e-05,
3.706492885600453e-05,
3.703905647252259e-05,
3.7211556946815896e-05,
3.7205256434579124e-05,
3.720713005122366e-05,
3.783155315246947e-05,
3.779755691519634e-05,
3.876107427433123e-05,
3.898063175867955e-05,
3.926808749130461e-05,
3.927262884968063e-05,
4.0621511414658814e-05,
4.06780122367672e-05,
4.087071570859418e-05,
4.0852279769809636e-05,
4.0793365427180154e-05,
4.17001724846645e-05,
4.238440950483778e-05,
4.2442585214696755e-05,
4.4623749288560034e-05,
4.462183693796243e-05,
4.5006692594700554e-05,
4.500525896414342e-05,
4.551165749699613e-05,
4.5618847783469284e-05,
4.5209486340352864e-05,
4.512785935622588e-05,
4.553247011952191e-05,
4.526373932840068e-05,
4.740394027066338e-05,
4.724829238601151e-05,
4.742326124075128e-05,
4.748027746158224e-05,
4.732667978878138e-05,
4.7556203756402955e-05,
4.7020401568329845e-05,
4.688760813887307e-05,
4.73688501549358e-05,
4.8204279390375e-05,
4.8326584455827476e-05,
4.838184579143742e-05,
4.8292552962752164e-05,
4.958400113830392e-05,
4.997055761082653e-05,
5.011110273192942e-05,
5.120280149244291e-05,
5.133951606273318e-05,
5.132804780876493e-05,
5.1390211851008654e-05,
5.1854594795421475e-05,
5.168904319231011e-05,
5.1796179567444495e-05,
```

```
5.176403386454182e-05,
5.1781365332321494e-05,
5.155308843989122e-05,
5.1294033232150756e-05,
5.132113767153607e-05,
5.129120786694492e-05,
5.134147347119458e-05,
5.122506292291152e-05,
5.111514876999936e-05,
5.194090352874217e-05,
5.1713672769240495e-05,
5.171500711439954e-05,
5.171778963511035e-05,
5.175761272370834e-05,
5.1741735755391133e-05,
5.182164231961045e-05,
5.1736117751217353e-05,
5.172415607411624e-05,
5.163261556946816e-05,
5.152704167457156e-05,
5.150019778030734e-05,
5.0987440080946055e-05,
5.1058867229494724e-05,
5.101524758110416e-05,
5.105498007968128e-05,
5.116080993486372e-05,
5.112470056282806e-05,
5.091180215645355e-05,
5.13953916081705e-05,
5.126616960728515e-05,
5.123841570227029e-05,
5.12489103901853e-05,
5.1064765066717265e-05,
5.10312515019288e-05,
5.0883568424713854e-05,
5.059787453361l6e-05,
5.0658356415607424e-05,
5.087342613672297e-05,
5.104218348826916e-05,
5.105039777398345e-05,
5.1056721684689826e-05,
5.083636169607287e-05,
5.089502735091382e-05,
5.089780734206034e-05,
5.106742743312466e-05,
4.9961849111490556e-05,
4.983181986340354e-05,
4.977205258331754e-05,
4.953270521090244e-05,
4.956255406943655e-05,
4.964731486751409e-05,
4.9643988332384756e-05,
4.938415275406312e-05,
4.946499652184912e-05,
4.8937529880478094e-05,
4.822388098400051e-05,
4.728214554480491e-05,
```

```
4.733597593751977e-05,
4.626680705748436e-05,
4.629018576487701e-05,
4.63135589388478e-05,
4.597443306140518e-05,
4.515380193511669e-05,
4.483913283374439e-05,
4.517835878707393e-05,
4.465565594763802e-05,
4.4603963036741925e-05,
4.5073709447922596e-05,
4.5066663346613545e-05,
4.5035256434579136e-05,
4.513720103712135e-05,
4.4978455068614426e-05,
4.49564933915133e-05,
4.510582606083601e-05,
4.504083523050654e-05,
4.5038104091570214e-05,
4.463589625624485e-05,
4.458755248845885e-05,
4.4473789129197486e-05,
4.4400862581420335e-05,
4.4393413488901525e-05,
4.530192547271231e-05,
4.489380889141844e-05,
4.522620170113197e-05,
4.514327120091063e-05,
4.454340289635109e-05,
4.48972951052931e-05,
4.4902900619743236e-05,
4.491037500790488e-05,
4.4878506766584445e-05,
4.4879799373932826e-05,
4.487002403086067e-05,
4.4894116233478775e-05,
4.484335736419401e-05,
4.502003541389995e-05,
4.497553784860557e-05,
4.5064514639853276e-05,
4.483449187377474e-05,
4.483936681844052e-05,
4.48761580661481e-05,
4.491092060330107e-05,
4.496581104154808e-05,
4.47408720672864e-05,
4.473096360589387e-05,
4.480141703029152e-05,
4.471725842661101e-05,
4.3722077246569275e-05,
4.3862606083602097e-05,
4.387556757098589e-05,
4.353238474672737e-05,
4.348472712325302e-05,
4.36684821033327e-05,
4.376419970910011e-05,
4.3578737747423e-05,
```

```
4.358016869031809e-05,
4.363180215645355e-05,
4.359843483210016e-05,
4.353408635300069e-05,
4.2542234237652556e-05,
4.277660073989755e-05,
4.324074416619237e-05,
4.321127237083412e-05,
4.2978704072598496e-05,
4.291319721115538e-05,
4.3038229779295516e-05,
4.315271659394169e-05,
4.315085878707393e-05,
4.283465123632455e-05,
4.2720916176563594e-05,
4.2720916176563594e-05,
4.225737051792829e-05,
4.23326369126668e-05,
4.239237763232784e-05,
4.261484474799216e-05,
4.267213858850313e-05,
4.236537706317587e-05,
4.233293540125213e-05,
4.2163214285714284e-05,
4.21615877758806e-05,
4.212432982356288e-05,
4.2017717700626056e-05,
4.1986094194649965e-05,
4.674894833364952e-05,
4.678378802251311e-05,
4.64060323784228e-05,
4.675382454309744e-05,
4.736170034149117e-05,
4.8411583507240865e-05,
4.8178922879908926e-05,
4.8226371972427736e-05,
4.8231031904129505e-05,
4.816899829254409e-05,
4.966482055903369e-05,
4.974091491178143e-05,
4.942849870359829e-05,
4.939606573705178e-05,
4.984594368557515e-05,
5.000415654840953e-05,
5.1052673591348875e-05,
5.0846572282299356e-05,
5.073260039208245e-05,
5.0720858787073913e-05,
5.0856817017643693e-05,
5.107087507114398e-05,
5.0978486055776884e-05,
5.06716997091001e-05,
5.069267865047745e-05,
5.093563460443938e-05,
5.0939132833744385e-05,
5.0874116707772076e-05,
5.079729020426231e-05,
```

```
5.0809668468981206e-05,
5.053291911718205e-05,
5.06688653323215e-05,
5.040071460190981e-05,
5.078503636248656e-05,
5.078429709732498e-05,
5.076652248150255e-05,
4.96885632074875e-05,
4.993666603427559e-05,
4.9949794947195336e-05,
4.9956782868525886e-05,
5.0223594985138805e-05,
5.014419069752735e-05,
5.0142559128565105e-05,
5.052763359261367e-05,
5.0531487225700365e-05,
5.07064812179852e-05,
5.0629794947195337e-05,
5.059475415797128e-05,
5.0565103870233345e-05,
5.018411354581673e-05,
5.0220472238031994e-05,
5.10322296528173e-05,
5.1399904508948325e-05,
5.064971668880035e-05,
4.9951964839056465e-05,
5.01179414089673e-05,
5.011209574400809e-05,
4.89855250426864e-05,
4.875260671599317e-05,
4.870785856573705e-05,
4.851628201479795e-05,
4.9028883039271484e-05,
4.914746015936255e-05,
4.848800654524758e-05,
4.858536457345222e-05,
4.756045089483336e-05,
4.734899007146018e-05,
4.680534797318661e-05,
4.69416733067729e-05,
4.5800455321570854e-05,
4.71509991778916e-05,
4.733509817871372e-05,
4.732014212989313e-05,
4.779181496237273e-05,
4.751791690381332e-05,
4.681719724277493e-05,
4.691303784860558e-05,
4.48333540441409e-05,
4.4722747581104155e-05,
4.496968443685575e-05,
4.5040062448618226e-05,
4.5106404224372356e-05,
4.510184199709101e-05,
4.519617466641372e-05,
4.539035271611965e-05,
4.494355324732815e-05,
```

```
4.48868253968254e-05,
4.3269764118130656e-05,
4.3297727660785436e-05,
4.309955163473092e-05,
4.2804291405805355e-05,
4.279939100739898e-05,
4.335406358692216e-05,
4.342488458862962e-05,
4.44799144691077e-05,
4.3940743059508003e-05,
4.3084080819578836e-05,
4.292945867324353e-05,
4.286928271042813e-05,
4.2926948713084175e-05,
4.214659077973819e-05,
4.15942672168469e-05,
4.2975019762220954e-05,
4.1865837443875286e-05,
4.409956033010813e-05,
4.504805207740465e-05,
4.518540109403654e-05,
4.504096044393852e-05,
4.508194823879086e-05,
4.5054006671725786e-05,
4.7721916619237326e-05,
4.8924486182255094e-05,
4.9051273477518485e-05,
4.9875940523619794e-05,
5.283148343135394e-05,
5.274575776260038e-05,
5.271121735281096e-05,
5.415944539303103e-05,
5.521627205463857e-05,
5.439987273129702e-05,
5.6356142888762394e-05,
5.6356929741352036e-05,
5.6343932840068285e-05,
5.690307515967873e-05,
5.6920813887307895e-05,
5.7339610921393776e-05,
5.7658002118510075e-05,
5.9949120976411795e-05,
6.164221716309364e-05,
6.169652943780432e-05,
6.223844479225953e-05,
6.267307199772338e-05,
6.305367276924049e-05,
6.309809081135773e-05,
6.363964001138301e-05,
6.354213795611204e-05,
6.419952618099029e-05,
6.417969771706821e-05,
6.390099032441658e-05,
6.382347878327954e-05,
6.395615948902798e-05,
6.721317586795671e-05,
6.731306187946622e-05,
```

```
6.738859988616956e-05,
6.737828542970968e-05,
6.737828542970968e-05,
6.730131078859161e-05,
6.718739692025543e-05,
6.710238474672731e-05,
6.714145133750707e-05,
6.781605182444818e-05,
6.775964886485798e-05,
6.781004916840568e-05,
6.782088582179214e-05,
6.777558843989117e-05,
6.775708388667546e-05,
6.791152390438241e-05,
6.790744877632322e-05,
6.803525121735274e-05,
6.8044528394359e-05,
6.799896161386194e-05,
6.820760940365515e-05,
6.817197811926888e-05,
6.806138873079104e-05,
7.057788702333515e-05,
7.059412066021619e-05,
7.096420208056653e-05,
7.093086368810464e-05,
7.094080108138864e-05,
7.15873964459621e-05,
7.154456776070312e-05,
7.145988031998979e-05,
7.209652105862255e-05,
7.489917077720854e-05,
7.454108644785926e-05,
7.474606889900706e-05,
7.534337870739255e-05,
7.7874116707772e-05,
7.787201463985319e-05,
7.878817476127229e-05,
7.951423306772899e-05,
8.253217352810969e-05,
8.267492110921385e-05,
8.334490782900137e-05,
8.33671422247517e-05,
8.480690887244664e-05,
8.486159109593365e-05,
8.582724656927836e-05,
8.683240039840628e-05,
8.850027746158215e-05,
8.907259849490915e-05,
9.155792259533286e-05,
9.153358423449052e-05,
9.136811025738307e-05,
9.14996754252829e-05,
9.16879546891797e-05,
9.211391513311823e-05,
9.300968870549539e-05,
9.300149291721991e-05,
9.316329665465114e-05,
```

```
9.58174932017959e-05,
9.591987905520764e-05,
9.590188515778147e-05,
9.664258821855424e-05,
9.746723755770558e-05,
9.761761509517474e-05,
9.750814013786114e-05,
9.764649386580649e-05,
9.877608976791236e-05,
9.901466562322128e-05,
9.892305761082642e-05,
9.963385315879327e-05,
9.942560472396118e-05,
9.960496426990439e-05,
0.0001000974822930499,
0.00010009169038133171,
0.00010275402327199128,
0.00010253495905267806,
0.00010369126478214115,
0.00010408373917030279,
0.00010434351530386379,
0.00010491965218491102,
0.00010521389663567938,
0.00010764504395117929,
0.00010780875924871928,
0.0001107712557705684,
0.00011081625434768849,
0.00011103405805350017,
0.00011252996901283741,
0.00011286241114905445,
0.00011307023762094466,
0.00011426460048694099,
0.00011443124818187554,
0.00011441155125529614,
0.00011516049452981711,
0.00011603070701321684,
0.00011634500268766191,
0.00012009265541010546,
0.00012032463858850298,
0.00012055900256118368,
0.00012057125577056836,
0.00012052831388730774,
0.00012063362296844352,
0.00012122146129766632,
0.00012168840495162192,
0.00012167632327831515,
0.00012167682207677213,
0.00012195795231771312,
0.00012250529864668295,
0.00012269929361917394,
0.00012360999984190206,
0.00012387624154176926,
0.00011921760371213541,
0.00011937408587870723,
0.00012207094336937947,
0.00012144388667551999,
0.00012094903244166177,
```

```
0.00012003992964649323,
0.00012194855293113245,
0.00012203108502497927,
0.0001233224111490544,
0.00012844216072219043,
0.0001270220317144 12,
0.00012712941883260 59,
0.00012861289492822344,
0.0001294382065389235,
0.00013530129134256606,
0.00013777478071839609,
0.0001384961651805475,
0.00013975391703029136,
0.00013987722427749304,
0.00014003227028394342,
0.00014030740593182808,
0.0001403603811737177,
0.00014042867387592474,
0.00014460827293998594,
0.0001445747680705747,
0.00014429290393979625,
0.00014430131790299107,
0.00014426313349775487,
0.0001442983298551823,
0.00014458457029026736,
0.00014458018639726796,
0.00014592551065578942,
0.00014588931685954583,
0.00014555285081894632,
0.00014791655647252252,
0.00014791978941377338,
0.00014901326440270652,
0.0001486710731676468,
0.00014899578622019848,
0.00014899940982103323,
0.00014875069800164412,
0.00014895162208309609,
0.00014894949519382776,
0.0001487253593562258,
0.00014872150382596588,
0.00014931920745589063,
0.00014930475178650466,
0.00014929860415480924,
0.00014956582100170737,
0.00015055191092771763,
0.00015058563634351473,
0.00014991944649971533,
0.00014997936571175605,
0.0001499530936571175,
0.00014997966989186107,
0.0001502597375577056,
0.00015028211471574014,
0.00015010101862391698,
0.00015019342566242958,
0.0001511194985138809,
0.00015115368794662 61,
0.00015117178697907 8,
```

```
0.00015111175045848347,
0.00015121310504015678,
0.00015102824448238783,
0.00015106478972996894,
0.00015126868525896408,
0.00015133297413520515,
0.00015122733383924613,
0.00015255254584835257,
0.0001540869820717131,
0.00015361870407259843,
0.00015369815009802057,
0.00015413839499146267,
0.00015333174160500848,
0.00015411071697337629,
0.00015393446958198944,
0.00015398840621640416,
0.0001542298847467273,
0.00015363360273192936,
0.0001546920660216277,
0.00015425375577056844,
0.00015430832527034712,
0.00015441617466641365,
0.0001567458551508252,
0.000156759671156643 2,
0.00015676047239612968,
0.00015623685116676145,
0.00015785387386960088,
0.0001578569891861126,
0.00015801536836779857,
0.00015925581531018774,
0.00015998269778030728,
0.00016012465123632448,
0.0001590704766647694,
0.00015924566163915755,
0.00015968669749573126,
0.00016215177180168205,
0.00016257118952760376,
0.00016245919306899376,
0.00016204018782014788,
0.0001625477118510086,
0.00016101809081135764,
0.00016326126098779477,
0.00016122057847973179,
0.00016143982719913987,
0.00016296534797318653,
0.00016371606320748742,
0.00016416098589767903,
0.00016409556187946617,
0.00016214179140580523,
0.00016329785176753294,
0.0001632619166192372,
0.00016333703408587 86,
0.00016205416050085362,
0.00016225012268386758,
0.00016259928713716553,
0.00016116211013090484,
0.00016029637307911202,
```

```
0.0001602936879466261,
0.0001586084171251501,
0.0001588938531587933,
0.0001590761639157654,
0.0001584326103522417 2,
0.0001588207568140136 7,
0.0001595195876810218 2,
0.0001593234958578383 6,
0.000157558968412065 9,
0.0001564969183583126 3,
0.000156966876620502,
0.0001565191505406942 3,
0.000156031006766584 3,
0.0001588811514260417 2,
0.0001588627844178838 5,
0.0001587880054385630 5,
0.0001635324130462276 2,
0.000163082456206918 2,
0.0001636133426611014 6,
0.000163505855150825 1,
0.0001716542457155503 4,
0.0001725610750648199 2,
0.0001770673665654839 3,
0.000184993877663947 2,
0.0001935176593625496 2,
0.0001986597495731358 6,
0.000199448856431417 17,
0.000199515186713463 4,
0.0002005584447922593 3,
0.0002004217800227658 9,
0.0002271595274457722 5,
0.0002354259348320999 6,
0.0002355064137102382 3,
0.0002433654183266930 5,
0.000244400201795990 62,
0.000250073425662429 4,
0.0002628446740024028 6,
0.0002628670250743057 7,
0.0002831747669638902 6,
0.000283527255264655 5,
0.0003378140536267625 6,
0.0003378313240688039 4,
0.000368714682065389,
0.0003755525384177573,
0.000375537357079617 8,
0.0003826719906089923 6,
0.000386121172453044 7,
0.0003986785927717698 4,
0.000398702162145070 3,
0.0004026045359830516 5,
0.000406467141434262 7,
0.0004441078081325489 5,
0.0004440423769999365 5,
0.000447660187820147 8,
0.0004468835458167328 7,
0.000446837515967874 3,
0.000453241454973755 5,
```

```
 0.00046204600818946416,
 0.00046849528157212394,
 0.00047162237273129683,
 0.0004714412856510464,
 0.00047287246569278 42,
 0.00048014850866375745,
 0.000499707996585088,
 0.0005060802276607853,
 0.0005079128059191803,
 0.000507008193258711,
 0.0005223826288496805,
 0.0005404577853664705,
 0.0005553016067476125,
 0.0005727887919749572,
 0.0005725248129703407,
 0.000575208387244672,
 0.0005757839706254346,
 0.0006063523879086826,
 0.0006126313885726932,
 0.0006152497508379181,
 0.0006174253176184151,
 0.0006220947751849743,
 0.0006292698632454309,
 0.0006306395261809903,
 0.0006295021684689812,
 0.000632521272370834,
 0.0006331266698286 22,
 0.0006331838335546701,
 0.0006442824174729652,
 0.0006440867556757098,
 0.0006455114431164232,
 ...],
1259)
```

In [140]:
```
var_e_q3_252 = np.zeros(shape=(1259,1))
for col_index in range(eqt_risk_prem_df.shape[1]):
    ri_minus_rf = eqt_risk_prem_df.iloc[:, col_index]
    rm_minus_rf = mkt_risk_prem_df[["Market Risk Premium"]]
    col_vars = []
    for i in range(1259):
        model = OLS(ri_minus_rf[i:i+251], add_constant(rm_minus_rf[i:i+2
51]))
        res = model.fit()
        varis = res.resid
        col_vars.append(np.var(varis))
    var_e_q3_252 = np.c_[var_e_q3_252, col_vars]
var_e_q3_252
```

Out[140]:
```
array([[0.        , 0.00039619, 0.00020445, ..., 0.00094436, 0.0005457
4,
        0.00077488],
       [0.        , 0.00039566, 0.00020412, ..., 0.00094415, 0.0005471
4,
        0.00077478],
       [0.        , 0.00039614, 0.00020326, ..., 0.0009464 , 0.0005350
6,
        0.00077391],
       ...,
       [0.        , 0.00027243, 0.00088567, ..., 0.0024766 , 0.0007879
2,
        0.00138948],
       [0.        , 0.00026971, 0.00088678, ..., 0.00248556, 0.0007875
1,
        0.00137423],
       [0.        , 0.00026425, 0.00087663, ..., 0.00248117, 0.0007877
3,
        0.001363  ]])
```

In [141]:
```
vars_df_q3_252 = pd.DataFrame(var_e_q3_252).drop(0, axis = 1)
vars_df_q3_252
```

Out[141]:

	1	2	3	4	5	6	7	8	9
0	0.000396	0.000204	0.000711	0.000472	0.000259	0.000552	0.001117	0.000185	0.000216
1	0.000396	0.000204	0.000707	0.000470	0.000259	0.000552	0.001101	0.000179	0.000218
2	0.000396	0.000203	0.000707	0.000481	0.000258	0.000551	0.001063	0.000177	0.000218
3	0.000393	0.000203	0.000699	0.000481	0.000257	0.000553	0.001029	0.000177	0.000218
4	0.000393	0.000203	0.000701	0.000481	0.000254	0.000552	0.001016	0.000177	0.000220
...	...	...	...	...	...	...	...	...	...
1254	0.000274	0.000891	0.002006	0.000372	0.000411	0.000232	0.000821	0.000233	0.002552
1255	0.000274	0.000888	0.002007	0.000371	0.000410	0.000230	0.000833	0.000233	0.002547
1256	0.000272	0.000886	0.002007	0.000371	0.000410	0.000233	0.000824	0.000232	0.002542
1257	0.000270	0.000887	0.002008	0.000370	0.000410	0.000233	0.000823	0.000232	0.002542
1258	0.000264	0.000877	0.002007	0.000371	0.000407	0.000231	0.000823	0.000231	0.002543

1259 rows × 100 columns

iii) Using the market capitalization weights (from the last day in the rolling window)of your securities, estimate the variance and standard deviation of your portfolio.

Formula:

$$\hat{\sigma}_p^2 = Var(\tilde{r}_p) = w^T \hat{\beta} \hat{\sigma}_M^2 \hat{\beta}^T w + w^T \hat{\Delta} w$$

In [142]:
```
df_weights_252 = df_weights.tail(1259)
df_weights_252
```

Out[142]:

Ticker #	1	2	3	4	5	6	7	8	9
251	0.014672	0.034202	0.001147	0.000248	0.004016	0.032580	0.000359	0.007714	0.000256
252	0.014634	0.033957	0.001150	0.000233	0.004031	0.032230	0.000350	0.007694	0.000253
253	0.014417	0.033738	0.001083	0.000234	0.004051	0.032946	0.000339	0.007724	0.000252
254	0.014450	0.033630	0.001037	0.000234	0.004070	0.033326	0.000330	0.007730	0.000244
255	0.014070	0.033624	0.001045	0.000230	0.004048	0.033202	0.000328	0.007761	0.000248
...	...	...	...	...	...	...	...	...	...
1505	0.013604	0.020282	0.000935	0.000440	0.005093	0.236809	0.000689	0.009864	0.000120
1506	0.013457	0.020469	0.000932	0.000440	0.005034	0.242061	0.000694	0.009756	0.000119
1507	0.013420	0.020067	0.000895	0.000433	0.004990	0.243796	0.000701	0.009732	0.000123
1508	0.013447	0.020065	0.000892	0.000441	0.004955	0.241941	0.000713	0.009759	0.000125
1509	0.013719	0.020300	0.000894	0.000437	0.004960	0.243645	0.000721	0.009688	0.000126

1259 rows × 100 columns

In [143]:
```
var_port_q3_252 = []
for i in range(len(df_weights_252)):
    diag_mat = np.diag(vars_df_q3_252.iloc[i])
    res = var_portfolio(df_weights_252.iloc[i], betas_df_q3_252.iloc[i],
var_rm_q3_252[i], diag_mat)
    var_port_q3_252.append(res)
arr_var_q3_252 = np.array(var_port_q3_252)
arr_var_q3_252
```

Out[143]:
```
array([7.10001196e-05, 7.15088950e-05, 7.13106077e-05, ...,
       2.80209391e-04, 2.79455380e-04, 2.78235687e-04])
```

In [144]:
```
arr_sd_q3_252 = np.sqrt(arr_var_q3_252)
arr_sd_q3_252
```

Out[144]:
```
array([0.00842616, 0.00845629, 0.00844456, ..., 0.01673946, 0.01671692,
       0.0166804 ])
```

iv) Using the market capitalization weights and returns (from the day following the last day in the rolling window) of your securities, calculate the one-day ahead return of the portfolio, $\bar{r}_p$.

In [145]:
```
#getting one-day ahead returns array
dayahead252_port_ret_q3 = []
dayahead252_ret_q3 = portfolio_q2_ret.loc[252:1510].to_numpy()
dayahead252_w_q3 = weights_q2_252.loc[252:1510].to_numpy()
for i in range(0, len(dayahead252_w_q2)):
    dayahead252_port_ret_q3.append(np.multiply(dayahead252_ret_q3[i], da
yahead252_w_q3[i]))
dayahead252_port_ret_q3_arr = np.sum(np.array(dayahead252_port_ret_q3),
axis = 1)
dayahead252_port_ret_q3_arr
```

Out[145]:
```
array([-0.00643605, -0.01146078, -0.00250408, ..., -0.00414611,
        0.00514476, -0.00946444])
```

v) Calculate the standardized outcome, z_p, where $z_p = \frac{\check{r}_p}{\hat{\sigma}_p}$ where we make the simplifying assumption that $E[\check{r}_p] = 0$.

In [146]:
```
standardized_outcomes_252_q3 = dayahead252_port_ret_q3_arr / arr_sd_q3_2
52
std_outcomes_252_q3 = pd.DataFrame(standardized_outcomes_252_q3)
std_outcomes_252_q3.index += 252
std_outcomes_252_q3.rename(columns={0: "Standardized Outcome"}, inplace
= True)
std_outcomes_252_q3
```

Out[146]:

	Standardized Outcome
252	-0.763818
253	-1.355296
254	-0.296531
255	0.545255
256	0.646344
...	...
1506	0.722503
1507	0.309746
1508	-0.247685
1509	0.307758
1510	-0.567399

1259 rows × 1 columns

Rolling Window 126

i) Use OLS to estimate the market betas for each stock:

In [148]:
```
betas_q3_126 = np.zeros(shape=(1385,1))
for col_index in range(eqt_risk_prem_df.shape[1]):
    ri_minus_rf = eqt_risk_prem_df.iloc[:, col_index]
    rm_minus_rf = mkt_risk_prem_df[["Market Risk Premium"]]
    col_beta = []
    for i in range(1385):
        model = OLS(ri_minus_rf[i:i+125], add_constant(rm_minus_rf[i:i+1
25]))
        res = model.fit()
        beta = res.params[1]
        col_beta.append(beta)
    betas_q3_126 = np.c_[betas_q3_126, col_beta]
betas_q3_126
```

Out[148]:
```
array([[0.        , 1.8587196 , 1.74873984, ..., 2.68095759, 1.5151042
9,
        2.36746127],
       [0.        , 1.85954524, 1.74715317, ..., 2.67584122, 1.5196995
8,
        2.37150663],
       [0.        , 1.87006592, 1.71116045, ..., 2.74035345, 1.6172243
2,
        2.33411603],
       ...,
       [0.        , 1.18005442, 1.8923816 , ..., 1.65167118, 0.9846047
7,
        2.01873032],
       [0.        , 1.181316  , 1.8953813 , ..., 1.62771547, 0.9696146
3,
        2.03000054],
       [0.        , 1.17989873, 1.89854685, ..., 1.6231458 , 0.9557467
7,
        2.03780855]])
```

In [149]:
```
betas_df_q3_126 = pd.DataFrame(betas_q3_126).drop(0, axis = 1)
betas_df_q3_126
```

Out[149]:

	1	2	3	4	5	6	7	8	9
0	1.858720	1.748740	2.090839	0.863111	0.797002	1.536315	2.082316	0.950925	1.132253
1	1.859545	1.747153	2.090576	0.861535	0.799008	1.546799	2.076093	0.945995	1.132293
2	1.870066	1.711160	2.140906	0.853506	0.832043	1.503603	2.209823	0.932673	1.136640
3	1.860781	1.719649	2.121365	0.852611	0.824514	1.499804	2.184585	0.932889	1.142616
4	1.855515	1.710480	2.139003	0.877110	0.857206	1.489830	2.152956	0.922099	1.178655
...	...	...	...	...	...	...	...	...	...
1380	1.177943	1.872456	1.285212	0.903495	0.548774	0.877056	1.121534	0.447239	1.635484
1381	1.178430	1.873308	1.284081	0.902287	0.547696	0.878505	1.132335	0.446102	1.637488
1382	1.180054	1.892382	1.269821	0.900144	0.550474	0.888848	1.134361	0.449499	1.629157
1383	1.181316	1.895381	1.273381	0.885996	0.552735	0.893454	1.139273	0.439229	1.626673
1384	1.179899	1.898547	1.262461	0.873377	0.551200	0.896687	1.144386	0.433780	1.616631

1385 rows × 100 columns

ii) Estimate the variance of the market, $\hat{\sigma}_M^2 = Var(\tilde{r}_M)$ and the idiosyncratic variance, $\hat{\sigma}_i^2 = Var(\tilde{\epsilon}_i)$, of each security in your portfolio.

Processing math: 100%

In [150]:
```
var_rm_q3_126 = list(ffdata['Market Returns'].rolling(126).var().dropna
())
var_rm_q3_126 = var_rm_q3_126[:-1]
var_rm_q3_126, len(var_rm_q3_126)
```

Out[150]: ([5.677102984126982e-05,
5.74354641269841e-05,
5.6288599999999975e-05,
5.6952635555555524e-05,
5.697653079365077e-05,
5.678806603174601e-05,
5.6477554285714266e-05,
5.631318412698411e-05,
5.623261714285712e-05,
5.597202603174601e-05,
5.596053079365077e-05,
5.6019149841269824e-05,
5.758051555555554e-05,
5.7309399365079346e-05,
5.803026857142855e-05,
5.8169119999999986e-05,
5.8134450793650776e-05,
5.759706412698411e-05,
5.6427509841269816e-05,
5.642999428571426e-05,
5.653655428571426e-05,
5.6796577142857116e-05,
5.6829034285714256e-05,
5.804937714285711e-05,
5.9706037460317435e-05,
5.795551555555553e-05,
5.94367441269841e-05,
5.918790603174601e-05,
5.923559936507934e-05,
5.9165399365079344e-05,
6.06546241269841e-05,
5.989579936507934e-05,
6.117797650793648e-05,
6.10408641269841e-05,
6.152750984126982e-05,
6.142502857142855e-05,
6.142585714285713e-05,
6.172903936507934e-05,
6.164881269841267e-05,
6.170429714285712e-05,
6.130849269841267e-05,
6.131277269841267e-05,
6.140513079365077e-05,
6.215309269841267e-05,
6.219962984126982e-05,
6.217807746031743e-05,
6.199473079365076e-05,
6.0256172698412666e-05,
5.886957269841266e-05,
5.7667226031745995e-05,
5.5720226031746e-05,
5.590139428571425e-05,
5.4812803174603136e-05,
5.487893650793647e-05,
5.4327744126984095e-05,
5.311295999999997e-05,
5.450391428571425e-05,

```
5.45097041269841e-05,
5.234939746031743e-05,
5.2720896507936474e-05,
5.1682412698412666e-05,
5.1650656507936476e-05,
5.1662808888888855e-05,
5.3046083174603144e-05,
5.255719428571426e-05,
5.221535746031744e-05,
5.250589714285712e-05,
5.3203954285714264e-05,
5.352573269841268e-05,
5.342370857142855e-05,
5.168130857142855e-05,
5.2025435555555534e-05,
5.255406984126982e-05,
5.25463041269841e-05,
5.266581650793649e-05,
5.1355812698412675e-05,
5.114575555555553e-05,
4.960228317460315e-05,
5.026799746031743e-05,
5.019603555555553e-05,
5.156333714285712e-05,
5.0782923174603144e-05,
5.0146243174603145e-05,
4.981625714285711e-05,
4.927307999999997e-05,
4.918681714285711e-05,
5.018415555555552e-05,
5.110594857142854e-05,
4.900903428571425e-05,
4.736958984126981e-05,
4.668670222222219e-05,
4.668497269841267e-05,
4.720925714285711e-05,
4.769302857142854e-05,
4.661413079365077e-05,
4.660521714285712e-05,
4.679145714285712e-05,
4.679414857142854e-05,
4.761261714285712e-05,
4.777371428571426e-05,
4.581520317460315e-05,
4.5941892698412675e-05,
4.575125079365077e-05,
4.581652317460315e-05,
4.5936097142857115e-05,
4.733638857142854e-05,
4.669751428571426e-05,
4.647058857142854e-05,
4.471250857142854e-05,
4.5892137142857116e-05,
4.497951746031743e-05,
4.501105269841267e-05,
4.393079428571426e-05,
4.399115936507934e-05,
```

```
4.409009079365077e-05,
4.4154197460317436e-05,
4.426144317460315e-05,
4.431371428571426e-05,
4.424946603174601e-05,
4.42627022222222e-05,
4.4271410793650774e-05,
4.421704888888887e-05,
4.432007428571427e-05,
4.471846603174601e-05,
4.4538290793650775e-05,
4.362635428571427e-05,
4.3597909841269826e-05,
4.3778679999999987e-05,
4.523687936507935e-05,
4.4822148571428555e-05,
4.4830709841269825e-05,
4.489570857142856e-05,
4.499239428571427e-05,
4.529851936507935e-05,
4.527061079365078e-05,
4.5497874285714275e-05,
4.572911746031744e-05,
4.593425079365078e-05,
4.483540317460316e-05,
4.537666857142856e-05,
4.465817650793649e-05,
4.468498603174602e-05,
4.3977026031746016e-05,
4.422154984126983e-05,
4.394322603174602e-05,
4.404022857142857e-05,
4.458621269841269e-05,
4.443646984126983e-05,
4.4429537142857134e-05,
4.224875936507936e-05,
4.093177714285714e-05,
4.097494857142857e-05,
3.982203428571428e-05,
4.0485514285714285e-05,
3.9270603174603175e-05,
3.960902857142857e-05,
3.8303548571428573e-05,
3.829802222222222e-05,
3.7188942222222225e-05,
3.758330412698413e-05,
3.71755326984127e-05,
3.8739759365079365e-05,
3.88658107936508e-05,
3.882186412698413e-05,
3.9446557460317465e-05,
3.978134857142858e-05,
3.9274520000000006e-05,
3.914633714285715e-05,
3.9098337142857154e-05,
3.895248317460318e-05,
3.87511726984127e-05,
```

```
3.864365714285715e-05,
3.9277117460317464e-05,
3.92454507936508e-05,
3.937840317460318e-05,
3.9524355555555555e-05,
3.9913217142857144e-05,
4.008658603174603e-05,
4.0012483174603174e-05,
3.999291428571428e-05,
3.99763326984127e-05,
4.029210412698413e-05,
3.8916874285714285e-05,
3.878610603174603e-05,
3.877953269841269e-05,
3.895995936507936e-05,
3.991415936507936e-05,
3.978090222222222e-05,
4.007958412698413e-05,
3.8477932698412696e-05,
3.8477932698412696e-05,
3.85037365079365e-05,
3.8351017142857134e-05,
3.8067748571428565e-05,
3.7671875555555547e-05,
3.781515428571427e-05,
3.897439555555555e-05,
3.956560317460317e-05,
4.109675428571428e-05,
4.099711999999999e-05,
4.1334217142857136e-05,
4.204116317460316e-05,
4.444918857142856e-05,
4.4785330793650785e-05,
4.458640317460316e-05,
4.525620317460316e-05,
4.400123428571427e-05,
4.394561269841268e-05,
4.470807428571427e-05,
4.487082412698411e-05,
4.4883692698412685e-05,
4.615430857142855e-05,
4.515409269841268e-05,
4.3583349841269824e-05,
4.3779937142857126e-05,
4.456578603174602e-05,
4.463703428571427e-05,
4.5560483174603156e-05,
4.513246984126983e-05,
4.5251509841269827e-05,
4.560587428571427e-05,
4.6323748571428555e-05,
4.6221714285714266e-05,
4.624154412698411e-05,
4.5574359999999986e-05,
4.529227746031745e-05,
4.5451744126984114e-05,
4.561078857142856e-05,
```

```
4.5640119999999985e-05,
4.575631999999998e-05,
4.626823428571427e-05,
4.4764149841269824e-05,
4.5025323174603164e-05,
4.503382603174602e-05,
4.5018892698412684e-05,
4.395207936507936e-05,
4.407079428571428e-05,
4.397014857142856e-05,
4.401643428571427e-05,
4.353485714285713e-05,
4.345867936507935e-05,
4.354710412698412e-05,
4.3566395555555544e-05,
4.340819555555555e-05,
4.343696317460317e-05,
4.2828759999999996e-05,
4.354989269841269e-05,
4.389851936507935e-05,
4.3774586031746017e-05,
4.4000292698412686e-05,
4.399915428571427e-05,
4.427949714285713e-05,
4.437746603174602e-05,
4.411423746031745e-05,
4.307554603174602e-05,
4.279383428571427e-05,
4.377911428571427e-05,
4.408898603174602e-05,
4.4013994285714275e-05,
4.364911746031745e-05,
4.353953269841268e-05,
4.3113528888888873e-05,
4.306919936507935e-05,
4.291274984126982e-05,
4.241293714285712e-05,
4.233935999999998e-05,
4.232987746031744e-05,
4.2271624126984106e-05,
4.225051428571426e-05,
4.20981841269841e-05,
4.243025269841267e-05,
4.2722497142857116e-05,
4.2107497142857116e-05,
4.213847428571426e-05,
4.205525079365076e-05,
4.249394857142854e-05,
4.2277666031746e-05,
4.2351563174603145e-05,
4.2548096507936476e-05,
4.184273269841267e-05,
4.209691999999997e-05,
4.190747428571425e-05,
4.194258857142854e-05,
4.301477079365076e-05,
4.3016639999999966e-05,
```

```
4.262571936507933e-05,
4.264576317460314e-05,
4.109467746031743e-05,
4.1021976507936475e-05,
4.077984317460314e-05,
4.0066603174603145e-05,
4.0159866031746004e-05,
4.022173714285712e-05,
4.0276508571428547e-05,
4.1324544126984104e-05,
4.066077714285712e-05,
4.073894857142855e-05,
4.163195428571426e-05,
4.090027999999998e-05,
4.101209650793648e-05,
4.120811936507933e-05,
4.1000068571428545e-05,
4.1091456507936484e-05,
4.067841079365077e-05,
4.0650839365079346e-05,
4.098000317460315e-05,
4.098000317460315e-05,
4.080164317460315e-05,
4.1509188571428544e-05,
4.1530052698412674e-05,
4.153684888888887e-05,
4.09679822222222e-05,
3.970829714285712e-05,
3.970628317460315e-05,
3.9937344126984105e-05,
3.9945346031746006e-05,
3.986758857142854e-05,
4.065009269841267e-05,
4.281505079365077e-05,
4.2558237460317436e-05,
4.270137714285712e-05,
4.299818984126982e-05,
4.190139428571426e-05,
4.143829650793648e-05,
3.938955936507934e-05,
3.998999428571426e-05,
3.959397079365077e-05,
3.926153714285712e-05,
3.831691746031744e-05,
3.971743746031744e-05,
3.921336888888886e-05,
4.075099428571426e-05,
4.079569079365077e-05,
3.9827690793650774e-05,
4.0374359365079344e-05,
4.188257079365078e-05,
4.2510504126984106e-05,
4.132157269841268e-05,
4.2356777142857126e-05,
4.245962222222221e-05,
4.2192544126984117e-05,
4.135691428571427e-05,
```

```
4.142957714285713e-05,
4.043727555555554e-05,
4.0483908571428554e-05,
3.989746857142855e-05,
3.962714984126982e-05,
3.891435428571427e-05,
3.876777650793649e-05,
3.954735746031745e-05,
3.951169269841269e-05,
3.9779883174603166e-05,
3.9761468571428566e-05,
3.9507617142857135e-05,
3.9498628571428563e-05,
4.0210314285714285e-05,
3.9646057142857143e-05,
3.980339555555555e-05,
4.0749897142857134e-05,
4.074215936507936e-05,
4.082311428571428e-05,
4.075922603174603e-05,
4.075531428571428e-05,
4.0720959999999996e-05,
4.071767936507935e-05,
4.067137079365078e-05,
4.072724888888888e-05,
4.061086222222221e-05,
4.061090603174602e-05,
4.0669828571428557e-05,
4.109709269841268e-05,
4.1096508571428556e-05,
4.031021269841268e-05,
4.002135936507935e-05,
4.001279999999998e-05,
4.022883555555554e-05,
4.0263456507936485e-05,
4.0000234285714266e-05,
4.012247936507934e-05,
4.160383555555554e-05,
4.142779428571427e-05,
4.1402502222222205e-05,
4.1044132698412685e-05,
4.07992565079365e-05,
4.076742603174602e-05,
4.081303936507935e-05,
4.120169714285713e-05,
4.120587746031745e-05,
4.110804317460316e-05,
4.0844315555555546e-05,
4.104050603174602e-05,
4.2783424126984117e-05,
4.261675999999999e-05,
4.255483936507935e-05,
4.2597346031746014e-05,
4.2941857142857126e-05,
4.293128888888888e-05,
4.2577923174603164e-05,
4.2594883174603165e-05,
```

```
4.223826603174602e-05,
4.2869224126984115e-05,
4.259555555555555e-05,
4.197974603174602e-05,
4.280278603174602e-05,
4.291986857142856e-05,
4.295393079365078e-05,
4.2562816507936496e-05,
4.254565079365078e-05,
4.315579428571428e-05,
4.207627746031745e-05,
4.2471989841269836e-05,
4.242497079365078e-05,
4.2498337142857135e-05,
4.2757795555555546e-05,
4.2757795555555546e-05,
4.257687428571428e-05,
4.261778857142857e-05,
4.307659746031746e-05,
4.343698857142857e-05,
4.337219555555555e-05,
4.2423154285714274e-05,
4.2916576507936496e-05,
4.3060599365079355e-05,
4.203005714285713e-05,
4.2158037460317446e-05,
4.2424959999999985e-05,
4.2030268571428554e-05,
4.2782610793650776e-05,
4.249170603174601e-05,
4.237195999999998e-05,
4.2371462222222206e-05,
4.205732317460316e-05,
4.229345269841268e-05,
4.200777714285713e-05,
4.122662857142856e-05,
4.123679746031745e-05,
4.125769650793649e-05,
4.151783428571427e-05,
4.209259936507936e-05,
4.2107548571428566e-05,
4.1691349841269836e-05,
4.16654126984127e-05,
4.185591428571428e-05,
4.114441714285714e-05,
3.868733714285714e-05,
3.9237963174603175e-05,
3.9123104126984126e-05,
3.924766984126984e-05,
3.915383428571429e-05,
3.8496457142857144e-05,
3.847440317460317e-05,
4.002945079365079e-05,
4.006562603174603e-05,
3.8958959365079364e-05,
3.7592737142857146e-05,
3.5725115555555556e-05,
```

```
3.588425714285714e-05,
3.3826732698412695e-05,
3.375648317460317e-05,
3.3513877460317455e-05,
3.227874857142857e-05,
3.090663428571428e-05,
3.0446450793650787e-05,
3.0440546031746026e-05,
3.0191622222222215e-05,
3.0110997460317452e-05,
3.0098669841269832e-05,
3.0128706031746024e-05,
3.164245079365079e-05,
3.3104972698412694e-05,
3.2801234285714276e-05,
3.2756919999999986e-05,
3.5235037460317445e-05,
3.550031999999999e-05,
3.5601117460317447e-05,
3.511691428571427e-05,
3.511961650793649e-05,
3.481039555555554e-05,
3.607748888888887e-05,
3.628781714285712e-05,
3.8159297142857125e-05,
3.7985948571428555e-05,
3.9178837460317446e-05,
3.904870857142856e-05,
4.067813079365078e-05,
4.078239746031744e-05,
4.1082079999999976e-05,
4.107801714285712e-05,
4.101482984126982e-05,
4.278845269841268e-05,
4.4074656507936486e-05,
4.427163746031744e-05,
4.8695319365079345e-05,
4.878098603174601e-05,
4.955114984126981e-05,
4.950758857142854e-05,
5.019842857142854e-05,
5.040007936507933e-05,
5.0295182222222194e-05,
5.0368170793650764e-05,
5.112015999999997e-05,
5.053793269841267e-05,
5.4860224126984096e-05,
5.481033650793648e-05,
5.508452888888886e-05,
5.363305650793648e-05,
5.3440064126984095e-05,
5.3906989841269803e-05,
5.315307999999996e-05,
5.314635936507933e-05,
5.408921714285711e-05,
5.5588599999999964e-05,
5.549866984126981e-05,
```

```
5.5568386031746e-05,
5.5512068571428544e-05,
5.857009079365076e-05,
5.904607999999997e-05,
5.766910222222219e-05,
6.009819746031743e-05,
6.041639428571426e-05,
6.0375426031746005e-05,
6.0086029841269816e-05,
6.105570603174601e-05,
6.1029344126984105e-05,
6.122702984126982e-05,
6.156854603174601e-05,
6.103999746031744e-05,
6.085847428571426e-05,
6.0900932698412674e-05,
6.0178732698412676e-05,
5.9960109841269816e-05,
6.005191428571426e-05,
6.0212584126984106e-05,
6.002551428571426e-05,
6.106930857142855e-05,
6.166271746031744e-05,
6.121035746031744e-05,
6.127189079365077e-05,
6.129599555555553e-05,
6.09624641269841e-05,
6.111183999999998e-05,
6.108096317460315e-05,
6.0988315555555535e-05,
6.0440759999999976e-05,
5.9822052698412675e-05,
5.9832429841269816e-05,
5.9822176507936486e-05,
5.954955746031743e-05,
5.934174984126982e-05,
6.044989714285712e-05,
6.054635428571426e-05,
6.017029079365077e-05,
6.0139448888888864e-05,
6.0327789841269815e-05,
6.031163746031744e-05,
6.033573079365078e-05,
6.0353948571428555e-05,
6.034987936507935e-05,
6.006902857142855e-05,
5.9999563174603156e-05,
6.015096888888887e-05,
6.022459936507935e-05,
6.069751428571427e-05,
6.0864736507936485e-05,
6.023515936507934e-05,
6.025250857142855e-05,
6.027186603174602e-05,
6.040379428571426e-05,
6.016973714285712e-05,
6.118759428571426e-05,
```

```
6.136968888888886e-05,
6.0616559365079345e-05,
6.064486603174601e-05,
6.0089199365079336e-05,
6.0201828571428545e-05,
6.10298641269841e-05,
6.106436317460315e-05,
5.888415999999997e-05,
5.8952704126984096e-05,
5.894685269841267e-05,
5.898138222222219e-05,
5.890187428571426e-05,
5.8949262222222195e-05,
5.892499555555553e-05,
5.90364241269841e-05,
5.932326603174601e-05,
5.9809626031746004e-05,
5.955081714285711e-05,
5.945349650793648e-05,
6.0048772698412665e-05,
5.919582984126981e-05,
5.919718857142854e-05,
6.020361650793648e-05,
6.0194586031746005e-05,
5.8720584126984094e-05,
5.751343999999997e-05,
5.751108317460315e-05,
5.7515210793650764e-05,
5.5295629841269815e-05,
5.4861994285714263e-05,
5.477281269841267e-05,
5.436560317460315e-05,
5.421154603174601e-05,
5.423867936507934e-05,
5.292299746031744e-05,
5.2743274285714265e-05,
5.261727746031744e-05,
5.207563746031744e-05,
5.1592617142857126e-05,
5.157209714285712e-05,
4.8574839365079347e-05,
4.908183936507935e-05,
4.869951555555554e-05,
4.869951555555554e-05,
4.870653714285712e-05,
4.668373079365078e-05,
4.506982603174601e-05,
4.5036422222222206e-05,
4.089147428571427e-05,
4.0998799365079354e-05,
3.997322984126983e-05,
4.025811555555554e-05,
3.935517269841269e-05,
3.9088408888888876e-05,
3.928210857142856e-05,
3.9365666031746024e-05,
3.851837714285713e-05,
```

```
3.875361269841269e-05,
3.4724210793650786e-05,
3.4961217142857134e-05,
3.4609234285714276e-05,
3.3905443174603164e-05,
3.441417714285713e-05,
3.387258984126983e-05,
3.3877217142857136e-05,
3.3802057142857136e-05,
3.2993399365079364e-05,
3.1311012698412694e-05,
3.097657714285714e-05,
3.081497714285714e-05,
3.110714222222222e-05,
2.8377984126984128e-05,
2.754235746031746e-05,
2.68532926984127e-05,
2.5128835555555556e-05,
2.592838857142857e-05,
2.5931083174603174e-05,
2.5593874285714284e-05,
2.460598603174603e-05,
2.4806834285714283e-05,
2.4700742222222218e-05,
2.445437714285714e-05,
2.446477714285714e-05,
2.4426709841269836e-05,
2.4298079999999996e-05,
2.4114199365079363e-05,
2.4413577142857138e-05,
2.4555908571428567e-05,
2.4685028571428567e-05,
2.4976937142857137e-05,
2.3479428571428563e-05,
2.29342965079365e-05,
2.2983189841269837e-05,
2.2969466031746027e-05,
2.2846877460317457e-05,
2.295501269841269e-05,
2.269463428571428e-05,
3.2525719999999994e-05,
3.2634799365079364e-05,
3.25572526984127e-05,
3.3915839999999994e-05,
3.5183692698412693e-05,
3.725083428571428e-05,
3.712389079365079e-05,
3.7436457142857144e-05,
3.624730984126984e-05,
3.598086857142858e-05,
3.943815428571429e-05,
3.957344317460318e-05,
3.87913326984127e-05,
3.877503746031746e-05,
3.9583424126984134e-05,
3.9863723174603176e-05,
4.1907317460317464e-05,
```

```
4.1775457142857144e-05,
4.1631212698412704e-05,
4.14088e-05,
4.15803307936508e-05,
4.166992000000001e-05,
4.138729650793652e-05,
4.138236000000001e-05,
4.141278984126985e-05,
4.182736317460319e-05,
4.165099746031748e-05,
4.16971688888889e-05,
4.062999936507938e-05,
4.051062412698414e-05,
4.0746748571428584e-05,
4.098354857142859e-05,
4.103286412698414e-05,
4.161718603174604e-05,
4.086113269841271e-05,
4.081446603174604e-05,
4.081300317460319e-05,
4.1154634285714297e-05,
4.1206834285714296e-05,
4.1208622222222235e-05,
4.179287936507937e-05,
4.164666984126985e-05,
4.166834857142858e-05,
4.2377704126984134e-05,
4.21192488888889e-05,
4.193182984126985e-05,
4.2004424126984136e-05,
4.203286857142858e-05,
4.126778412698413e-05,
4.130341650793651e-05,
4.13498926984127e-05,
4.21521326984127e-05,
4.287813650793652e-05,
4.292534857142858e-05,
4.2745224126984134e-05,
4.307570857142858e-05,
4.3074037460317476e-05,
4.306443936507938e-05,
4.303305714285716e-05,
4.303255936507938e-05,
4.305349079365081e-05,
4.422180317460319e-05,
4.443277269841271e-05,
4.443642412698414e-05,
4.481483746031748e-05,
4.286846222222224e-05,
4.29714088888889e-05,
4.2371226031746046e-05,
4.2674337142857155e-05,
4.339222603174604e-05,
4.553478412698414e-05,
4.6276523174603183e-05,
4.625302857142858e-05,
4.71702526984127e-05,
```

```
4.86326907936508e-05,
4.8794034285714297e-05,
4.9082062222222236e-05,
4.91043688888889e-05,
4.872980317460319e-05,
5.016405650793652e-05,
5.008098857142859e-05,
5.1107228571428586e-05,
5.1330588571428586e-05,
5.13450088888889e-05,
5.172913079365081e-05,
5.1625496507936525e-05,
5.1231474285714304e-05,
5.213057269841272e-05,
5.19706742857143e-05,
5.193359746031748e-05,
5.204173269841272e-05,
5.1525354285714303e-05,
5.317102984126985e-05,
5.3314250793650805e-05,
5.551307746031747e-05,
5.5229097142857156e-05,
5.512989269841271e-05,
5.513457079365081e-05,
5.516158222222223e-05,
5.503575936507938e-05,
5.6196902222222234e-05,
5.593089269841271e-05,
5.924591555555557e-05,
5.875776317460319e-05,
6.246841714285716e-05,
6.429873079365081e-05,
6.493063936507938e-05,
6.558310412698415e-05,
6.541095936507939e-05,
6.538287746031749e-05,
7.082060000000003e-05,
7.339146412698415e-05,
7.370138222222225e-05,
7.553977714285717e-05,
8.144231428571431e-05,
8.10435231746032e-05,
8.091242412698416e-05,
8.336612317460321e-05,
8.488923428571432e-05,
8.48922774603175e-05,
8.967485714285718e-05,
8.959262222222227e-05,
8.962933650793654e-05,
9.09596374603175e-05,
9.09235993650794e-05,
9.209397079365083e-05,
8.319092317460322e-05,
8.755578984126989e-05,
9.115941650793655e-05,
8.995859555555559e-05,
8.978070412698416e-05,
```

```
8.85963485714286e-05,
8.948685269841274e-05,
8.925914412698416e-05,
9.147990603174606e-05,
9.150384317460321e-05,
8.947435428571432e-05,
8.92929885714286e-05,
8.952106222222225e-05,
8.938016000000003e-05,
8.882573269841272e-05,
9.509987555555558e-05,
9.32285485714286e-05,
9.349404000000003e-05,
9.36324831746032e-05,
9.383159428571431e-05,
9.352291555555558e-05,
9.324077269841274e-05,
9.335422984126986e-05,
9.34376285714286e-05,
9.47595485714286e-05,
9.422966603174605e-05,
9.449374984126986e-05,
9.446138412698414e-05,
9.546245650793653e-05,
9.554319746031747e-05,
9.561750857142858e-05,
9.537437714285717e-05,
9.557663746031749e-05,
9.498351555555557e-05,
9.56482742857143e-05,
9.610458222222223e-05,
9.605894603174604e-05,
9.54448742857143e-05,
0.00010037401714285715,
0.00010043386412698414,
0.00010055601269841272,
0.00010064961269841272,
0.00010065357714285716,
0.00010134766222222224,
0.00010153195746031748,
0.0001014811942857143,
0.00010270420888888892,
0.00010818370603174605,
0.00010816710603174606,
0.00010852090412698415,
0.00010970317714285718,
0.00011404384000000002,
0.00011320895746031748,
0.00011496321079365081,
0.00011649878412698414,
0.00012233270412698416,
0.00012260497269841274,
0.0001238110685714286,
0.00012390337079365083,
0.0001266339307936508,
0.00012677514412698414,
0.00012761049650793655,
```

```
0.00012947030603174605,
0.00013261841079365084,
0.00013333435746031748,
0.00014037702603174606,
0.00014015780317460 32,
0.00014053691428571433,
0.00014055381079365085,
0.00014028038412698 42,
0.00013932825714285 72,
0.00014037828888888894,
0.00014037233079365084,
0.00013986096000000007,
0.00014352075428571435,
0.00014354729714285 72,
0.00014311174857142 86,
0.00014450212317460321,
0.00014643878412698416,
0.00014553908317460 32,
0.00014529258984126 99,
0.00014473377269841277,
0.00014689941079365084,
0.00014740897269841275,
0.00014676329714285 72,
0.00014824514603174608,
0.00014827305714285719,
0.00014757626412698416,
0.00014856410984126985,
0.00014855289650793653,
0.00015348375428571429,
0.00015345492317460 32,
0.00015426250412698414,
0.00015490637714285716,
0.00015296752,
0.00015425477079365082,
0.00015520806603174607,
0.00015980457714285718,
0.00016005617269841275,
0.00016553835428571433,
0.00016479121079365086,
0.00016519082412698 42,
0.00016574204000000006,
0.00016706637714285 72,
0.00016399547428571433,
0.00016489594412698 42,
0.00016452666412698 42,
0.00016395313079365086,
0.00016569050222222227,
0.00016750651428571435,
0.00016279551746031752,
0.00016755344000000006,
0.00016759809079365084,
0.00016608326857142863,
0.00016051406222222228,
0.00016082662603174 61,
0.00016118493714285 72,
0.00016004598412698419,
0.00015944668000000004,
```

Processing math: 100%

```
0.00015942231936507938,
0.00015462605269841274,
0.00015528724317460322,
0.0001563208431746032,
0.00015532381269841273,
0.0001572362488888889,
0.00015654601714285715,
0.00015601802222222223,
0.00015211607428571426,
0.00015361633269841268,
0.00015352216888888886,
0.0001523668526984127,
0.00015173998222222222,
0.00015465334603174602,
0.00015494304317460316,
0.00015547015428571428,
0.00016607837714285715,
0.00016481232317460317,
0.00016532642603174602,
0.00016777579936507936,
0.0001693270907936508,
0.00018230444000000003,
0.00018015321714285714,
0.0001838062831746032,
0.00018636715746031746,
0.00018645348317460316,
0.00018659639936507936,
0.00018730335999999996,
0.00018739201269841267,
0.00018739201269841267,
0.00019612511555555553,
0.00019447295999999996,
0.00019455595936507933,
0.00019445465650793647,
0.00019440831999999997,
0.0001931903041269841,
0.00019368197079365075,
0.00019339771999999996,
0.00019617385650793647,
0.00019605185079365076,
0.00019603195999999995,
0.00020059063936507933,
0.00020026736317460311,
0.0002025811174603174,
0.00020257460888888884,
0.000198558626031746,
0.00019846983936507 93,
0.00019781218857142853,
0.00019819940317460 31,
0.00019819947746031738,
0.0001967637542857142,
0.00019648037650793642,
0.00019801563428571 42,
0.00019662567428571 42,
0.00019145189079365 07,
0.00019203466222222215,
0.00019364439936507 93,
```

```
 0.00019244610603174597,
 0.00018691849714285707,
 0.00018792781714285707,
 0.00018614247428571423,
 0.00018462909269841265,
 0.00017938770603174596,
 0.00017915967999999994,
 0.00017753443555555555,
 0.00017765530222222213,
 0.00017672281714285707,
 0.00017666922984126975,
 0.00017584466603174596,
 0.00017392495746031738,
 0.00017079350412698406,
 0.00016976578603174593,
 0.00016295078603174596,
 0.0001635827631746031,
 0.0001633395707936507,
 0.00016311038984126978,
 0.00016601934222222216,
 0.00017001325714285706,
 0.0001679476799999999,
 0.00016809844317460307,
 0.00016925355428571418,
 0.00016430509714285705,
 0.0001658656507936507,
 0.000165970532698412 6,
 0.00016470645650793 64,
 0.00016323758984126974,
 0.00016295718603174592,
 0.00016532523746031735,
 ...],
1385)
```

In [151]:
```
var_e_q3_126 = np.zeros(shape=(1385,1))
for col_index in range(eqt_risk_prem_df.shape[1]):
    ri_minus_rf = eqt_risk_prem_df.iloc[:, col_index]
    rm_minus_rf = mkt_risk_prem_df[["Market Risk Premium"]]
    col_vars = []
    for i in range(1385):
        model = OLS(ri_minus_rf[i:i+125], add_constant(rm_minus_rf[i:i+1
25]))
        res = model.fit()
        varis = res.resid
        col_vars.append(np.var(varis))
    var_e_q3_126 = np.c_[var_e_q3_126, col_vars]
var_e_q3_126
```

Out[151]:
```
array([[0.        , 0.00033091, 0.00023457, ..., 0.00089622, 0.0006101
7,
        0.00083786],
       [0.        , 0.00033016, 0.00023392, ..., 0.00089916, 0.0006110
3,
        0.00083976],
       [0.        , 0.0003314 , 0.00023348, ..., 0.00089376, 0.0005843
4,
        0.00083603],
       ...,
       [0.        , 0.00014199, 0.00045503, ..., 0.00159275, 0.0005720
9,
        0.00125177],
       [0.        , 0.00014195, 0.0004573 , ..., 0.00159915, 0.0005670
3,
        0.00124892],
       [0.        , 0.00014161, 0.00045658, ..., 0.00159687, 0.0005547
2,
        0.0012489 ]])
```

In [152]:
```
vars_df_q3_126 = pd.DataFrame(var_e_q3_126).drop(0, axis = 1)
vars_df_q3_126
```

Out[152]:

	1	2	3	4	5	6	7	8	9
0	0.000331	0.000235	0.000796	0.000553	0.000191	0.000413	0.001187	0.000131	0.000167
1	0.000330	0.000234	0.000791	0.000548	0.000191	0.000424	0.001148	0.000119	0.000166
2	0.000331	0.000233	0.000789	0.000550	0.000188	0.000421	0.001074	0.000118	0.000166
3	0.000325	0.000237	0.000765	0.000549	0.000188	0.000425	0.001007	0.000118	0.000171
4	0.000325	0.000237	0.000766	0.000551	0.000190	0.000423	0.000982	0.000118	0.000177
...	...	...	...	...	...	...	...	...	...
1380	0.000145	0.000470	0.001286	0.000267	0.000243	0.000147	0.000671	0.000142	0.001309
1381	0.000142	0.000470	0.001285	0.000267	0.000243	0.000145	0.000701	0.000141	0.001309
1382	0.000142	0.000455	0.001271	0.000266	0.000242	0.000150	0.000702	0.000140	0.001305
1383	0.000142	0.000457	0.001280	0.000262	0.000242	0.000150	0.000703	0.000138	0.001314
1384	0.000142	0.000457	0.001265	0.000253	0.000242	0.000151	0.000698	0.000134	0.001310

1385 rows × 100 columns

iii) Using the market capitalization weights (from the last day in the rolling window)of your securities, estimate the variance and standard deviation of your portfolio.

Formula:

$$\hat{\sigma}_p^2 = Var(\tilde{r}_p) = w^T \hat{\beta} \hat{\sigma}_M^2 \hat{\beta}^T w + w^T \hat{\Delta} w$$

In [153]:
```
df_weights_126 = df_weights.tail(1385)
df_weights_126
```

Out[153]:

Ticker #	1	2	3	4	5	6	7	8	9
125	0.018116	0.037240	0.001599	0.000318	0.004287	0.015935	0.000463	0.008304	0.000257
126	0.017800	0.037490	0.001569	0.000324	0.004315	0.016075	0.000474	0.008177	0.000257
127	0.017727	0.038398	0.001550	0.000328	0.004267	0.015777	0.000474	0.008146	0.000264
128	0.017547	0.038386	0.001501	0.000322	0.004119	0.015748	0.000472	0.008176	0.000255
129	0.017530	0.038236	0.001515	0.000323	0.004116	0.015692	0.000479	0.008088	0.000256
...	...	...	...	...	...	...	...	...	...
1505	0.013604	0.020282	0.000935	0.000440	0.005093	0.236809	0.000689	0.009864	0.000120
1506	0.013457	0.020469	0.000932	0.000440	0.005034	0.242061	0.000694	0.009756	0.000119
1507	0.013420	0.020067	0.000895	0.000433	0.004990	0.243796	0.000701	0.009732	0.000123
1508	0.013447	0.020065	0.000892	0.000441	0.004955	0.241941	0.000713	0.009759	0.000125
1509	0.013719	0.020300	0.000894	0.000437	0.004960	0.243645	0.000721	0.009688	0.000126

1385 rows × 100 columns

In [154]:
```
var_port_q3_126 = []
for i in range(len(df_weights_126)):
    diag_mat = np.diag(vars_df_q3_126.iloc[i])
    res = var_portfolio(df_weights_126.iloc[i], betas_df_q3_126.iloc[i],
var_rm_q3_126[i], diag_mat)
    var_port_q3_126.append(res)
arr_var_q3_126 = np.array(var_port_q3_126)
arr_var_q3_126
```

Out[154]:
```
array([6.97628889e-05, 7.02152997e-05, 6.97454828e-05, ...,
       1.08765762e-04, 1.07983400e-04, 1.08278851e-04])
```

In [155]:
```
arr_sd_q3_126 = np.sqrt(arr_var_q3_126)
arr_sd_q3_126
```

Out[155]:
```
array([0.00835242, 0.00837946, 0.00835138, ..., 0.01042908, 0.01039151,
       0.01040571])
```

iv) Using the market capitalization weights and returns (from the day following the last day in the rolling window) of your securities, calculate the one-day ahead return of the portfolio, $\bar{r}_p$.

In [156]:
```
#getting one-day ahead returns array
dayahead126_port_ret_q3 = []
dayahead126_ret_q3 = portfolio_q2_ret.loc[126:1510].to_numpy()
dayahead126_w_q3 = weights_q2_126.loc[126:1510].to_numpy()
for i in range(0, len(dayahead126_w_q2)):
    dayahead126_port_ret_q3.append(np.multiply(dayahead126_ret_q3[i], da
yahead126_w_q3[i]))
dayahead126_port_ret_q3_arr = np.sum(np.array(dayahead126_port_ret_q3),
axis = 1)
dayahead126_port_ret_q3_arr
```

Out[156]:
```
array([-0.01262911,  0.00056176, -0.0061875 , ..., -0.00414611,
        0.00514476, -0.00946444])
```

v) Calculate the standardized outcome, z_p, where $z_p = \frac{\tilde{r}_p}{\hat{\sigma}_p}$ where we make the simplifying assumption that $E[\tilde{r}_p] = 0$.

In [158]:
```
standardized_outcomes_126_q3 = dayahead126_port_ret_q3_arr / arr_sd_q3_1
26
std_outcomes_126_q3 = pd.DataFrame(standardized_outcomes_126_q3)
std_outcomes_126_q3.index += 126
std_outcomes_126_q3.rename(columns={0: "Standardized Outcome"}, inplace
= True)
std_outcomes_126_q3
```

Out[158]:

	Standardized Outcome
126	-1.512030
127	0.067040
128	-0.740896
129	-0.016961
130	0.009658
...	...
1506	1.164179
1507	0.498764
1508	-0.397552
1509	0.495093
1510	-0.909543

1385 rows × 1 columns

Rolling Window 63

i) Use OLS to estimate the market betas for each stock:

In [159]:
```
betas_q3_63 = np.zeros(shape=(1448,1))
for col_index in range(eqt_risk_prem_df.shape[1]):
    ri_minus_rf = eqt_risk_prem_df.iloc[:, col_index]
    rm_minus_rf = mkt_risk_prem_df[["Market Risk Premium"]]
    col_beta = []
    for i in range(1448):
        model = OLS(ri_minus_rf[i:i+62], add_constant(rm_minus_rf[i:i+62
]))
        res = model.fit()
        beta = res.params[1]
        col_beta.append(beta)
    betas_q3_63 = np.c_[betas_q3_63, col_beta]
betas_q3_63
```

Out[159]:
```
array([[0.        , 2.05281222, 1.6314848 , ..., 2.75372606, 1.2764953
8,
        2.98277006],
       [0.        , 2.02427953, 1.61056927, ..., 2.7743287 , 1.3806743
3,
        3.00345426],
       [0.        , 2.02989276, 1.61751051, ..., 2.95940282, 1.5551235
9,
        2.94348151],
       ...,
       [0.        , 1.17241229, 1.90870263, ..., 2.19046177, 0.8616258
8,
        2.04912328],
       [0.        , 1.17280462, 1.90645963, ..., 2.19202489, 0.8599611
6,
        2.04474499],
       [0.        , 1.17361875, 1.90209361, ..., 2.19155911, 0.8424587
6,
        2.10855709]])
```

In [160]:
```
betas_df_q3_63 = pd.DataFrame(betas_q3_63).drop(0, axis = 1)
betas_df_q3_63
```

Out[160]:

	1	2	3	4	5	6	7	8	9
0	2.052812	1.631485	2.467977	0.149854	0.834601	1.489859	2.200903	0.891679	1.255516
1	2.024280	1.610569	2.470555	0.134584	0.813522	1.469599	2.103833	0.854639	1.232819
2	2.029893	1.617511	2.626662	0.150088	0.836575	1.398232	2.441186	0.785010	1.221262
3	2.002958	1.605841	2.581234	0.202040	0.870995	1.430943	2.490066	0.782768	1.185146
4	1.986856	1.585203	2.553937	0.221884	0.851413	1.438855	2.489432	0.773026	1.202064
...	...	...	...	...	...	...	...	...	...
1443	1.155123	1.901754	1.162571	0.910696	0.745226	0.835353	1.144480	0.652799	1.932211
1444	1.152748	1.883088	1.184261	0.903087	0.751742	0.834111	1.189235	0.650191	1.972411
1445	1.172412	1.908703	1.270773	0.916972	0.768153	0.839263	1.201479	0.652601	2.109775
1446	1.172805	1.906460	1.270128	0.914495	0.766944	0.837897	1.198084	0.654865	2.110638
1447	1.173619	1.902094	1.272323	0.905755	0.772203	0.841758	1.190653	0.665431	2.118114

1448 rows × 100 columns

ii) Estimate the variance of the market, $\hat{\sigma}_M^2 = Var(\tilde{r}_M)$ and the idiosyncratic variance, $\hat{\sigma}_i^2 = Var(\tilde{\epsilon}_i)$, of each security in your portfolio.

In [161]:
```
var_rm_q3_63 = list(ffdata['Market Returns'].rolling(63).var().dropna())
var_rm_q3_63 = var_rm_q3_63[:-1]
var_rm_q3_63, len(var_rm_q3_63)
```

Out[161]: ([5.8434905273937535e-05,
5.943506912442396e-05,
5.7831889400921654e-05,
5.80658064516129e-05,
5.8346810035842297e-05,
5.8113179723502306e-05,
5.7599262672811055e-05,
6.097627240143369e-05,
6.08163338453661e-05,
5.99999897593446e-05,
6.034555043522785e-05,
5.966281105990783e-05,
6.335046594982078e-05,
6.33131797235023e-05,
6.664884792626728e-05,
6.661854582693292e-05,
6.496304147465438e-05,
6.383350742447518e-05,
6.465539170506913e-05,
6.58262263184844e-05,
6.644691244239632e-05,
6.75279160266257e-05,
6.774866359447005e-05,
6.626358422939068e-05,
6.729861751152074e-05,
6.774078341013825e-05,
7.048802355350742e-05,
7.163447004608295e-05,
6.972407578084998e-05,
6.950336405529954e-05,
6.917300051203277e-05,
6.885972862263185e-05,
6.992598054275473e-05,
6.960167946748591e-05,
6.95037788018433e-05,
6.962149513568868e-05,
6.992038914490527e-05,
7.396512032770097e-05,
7.394442396313364e-05,
7.446379928315413e-05,
7.26520942140297e-05,
7.225603686635945e-05,
7.238912954429082e-05,
7.30578801843318e-05,
7.355756272401433e-05,
7.662930875576037e-05,
7.59322017409114e-05,
7.408658986175114e-05,
7.102296979006654e-05,
7.050626728110597e-05,
6.749326164874552e-05,
6.720955965181771e-05,
6.479865335381464e-05,
6.482390681003583e-05,
6.355966717869943e-05,
6.038661034306196e-05,
6.145529953917051e-05,

```
6.130404505888377e-05,
5.7231664106502824e-05,
5.734442396313365e-05,
5.477504352278546e-05,
5.482429083461343e-05,
5.646370711725551e-05,
5.580129032258065e-05,
5.575983614951357e-05,
5.5069953917050694e-05,
5.61678853046595e-05,
5.616020481310805e-05,
5.6124260112647214e-05,
5.583317972350231e-05,
5.247382488479263e-05,
5.2463937532002054e-05,
5.277248847926268e-05,
5.233700460829494e-05,
5.3209528929851515e-05,
5.273702508960574e-05,
5.2220496671787e-05,
5.0018617511520746e-05,
5.0253599590373785e-05,
5.213585765488991e-05,
5.208273937532002e-05,
4.907697900665643e-05,
4.79415565796211e-05,
4.7486359447004624e-05,
4.698124935995905e-05,
4.681345622119816e-05,
5.055940604198669e-05,
5.2904045058883776e-05,
4.909296979006657e-05,
4.902410138248849e-05,
4.7559943676395296e-05,
4.9595622119815674e-05,
4.9681356886840766e-05,
5.2916840757808506e-05,
5.120451612903227e-05,
5.285402969790067e-05,
5.300221198156683e-05,
5.4022570404505895e-05,
5.39436917562724e-05,
5.3732324628776246e-05,
5.044286226318485e-05,
5.031476702508961e-05,
4.992919098822325e-05,
5.0939354838709686e-05,
5.135160266257042e-05,
5.14067741935484e-05,
5.2063000512032785e-05,
5.1765550435227864e-05,
4.872379928315413e-05,
4.900949308755761e-05,
4.738533538146442e-05,
4.757981566820277e-05,
4.5739559651817724e-05,
4.480714797747057e-05,
```

```
4.543402969790067e-05,
4.570866359447006e-05,
4.5813364055299555e-05,
4.594874551971328e-05,
4.659704045058886e-05,
4.798451612903228e-05,
4.810248847926269e-05,
4.810768561187918e-05,
4.872404505888378e-05,
4.9325857654889924e-05,
4.920539170506914e-05,
4.7688545826932936e-05,
5.110544290834615e-05,
4.990608806963647e-05,
4.9880358422939085e-05,
4.902349206349208e-05,
5.077026625704047e-05,
5.1559308755760385e-05,
5.1591935483870985e-05,
5.15646543778802e-05,
5.230383000512034e-05,
5.305221710189453e-05,
5.336972862263186e-05,
5.281919098822326e-05,
5.06052124935996e-05,
5.063788018433181e-05,
4.9078453661034316e-05,
5.01899436763953e-05,
4.8604997439836154e-05,
5.1163215565796216e-05,
5.2877327188940094e-05,
5.287092165898618e-05,
5.278096774193549e-05,
5.191555043522786e-05,
5.178059907834102e-05,
4.898668202764978e-05,
4.730166410650283e-05,
4.7337470558115725e-05,
4.532383000512033e-05,
4.495499743983615e-05,
4.211770097286227e-05,
4.2939206349206355e-05,
4.1364290834613424e-05,
4.136966717869945e-05,
4.0270312339989775e-05,
4.024958525345623e-05,
3.957329237071174e-05,
4.1274674859191e-05,
4.162564260112648e-05,
4.093745519713263e-05,
4.111797235023043e-05,
4.108184331797236e-05,
3.9815442908346145e-05,
3.997372247823862e-05,
4.2469308755760375e-05,
4.1284290834613425e-05,
4.0772309267793146e-05,
```

```
4.028407578084998e-05,
4.2422073732718906e-05,
4.2492165898617515e-05,
4.240536610343063e-05,
4.237740911418332e-05,
4.322111111111112e-05,
4.2413492063492066e-05,
4.2380496671787e-05,
4.265456733230927e-05,
4.259856118791604e-05,
4.204267793138762e-05,
3.968896057347671e-05,
3.9341664106502824e-05,
3.9346682027649776e-05,
3.87162877624168e-05,
3.898902201740912e-05,
3.884920634920635e-05,
3.886189452124937e-05,
3.61225806451613e-05,
3.808533538146442e-05,
4.115435227854583e-05,
4.133747055811572e-05,
3.946450588837686e-05,
3.869354838709678e-05,
3.884175115207373e-05,
3.953504352278546e-05,
3.853990783410139e-05,
3.8175734767025096e-05,
3.838349206349207e-05,
3.9224101382488494e-05,
3.932175115207375e-05,
4.052603686635946e-05,
4.0868975934459815e-05,
3.9741244239631345e-05,
3.975601126472095e-05,
3.7887695852534575e-05,
3.570267793138762e-05,
3.591322580645163e-05,
3.707983614951358e-05,
3.7648294930875596e-05,
3.776271889400923e-05,
3.6105734767025104e-05,
3.478100358422941e-05,
3.46883461341526e-05,
3.463218637992833e-05,
3.613977470558117e-05,
3.638220174091143e-05,
3.608198668714799e-05,
3.496810035842295e-05,
3.505124423963134e-05,
3.435052739375321e-05,
3.479931387608808e-05,
3.479931387608808e-05,
3.629713261648746e-05,
3.608026625704046e-05,
3.6907731694828476e-05,
3.797479774705582e-05,
```

```
3.858497695852535e-05,
3.885737839221711e-05,
3.861608806963646e-05,
3.5632165898617516e-05,
3.690413722478239e-05,
3.6949308755760374e-05,
3.729618023553508e-05,
3.654413722478239e-05,
3.6350158730158726e-05,
3.659511520737327e-05,
3.704027649769585e-05,
3.667273937532001e-05,
3.737627240143368e-05,
3.723677419354838e-05,
3.710716333845365e-05,
3.7121280081925235e-05,
3.832225806451612e-05,
3.7052969790066554e-05,
3.682709677419354e-05,
3.6818484383000505e-05,
3.7581280081925234e-05,
3.934958525345622e-05,
3.937006656426011e-05,
3.977705069124424e-05,
3.987101894521249e-05,
3.859221198156682e-05,
3.6071280081925243e-05,
3.5855304659498206e-05,
3.686616487455197e-05,
3.6793906810035846e-05,
3.690958525345622e-05,
3.8529452124936e-05,
4.019028161802355e-05,
4.3390281618023555e-05,
4.2855320020481315e-05,
4.253382488479263e-05,
4.376336917562724e-05,
4.8274101382488476e-05,
4.8738131080389144e-05,
4.963128008192524e-05,
5.071152073732719e-05,
5.0092857142857147e-05,
5.215797235023042e-05,
5.377691244239632e-05,
5.2751105990783414e-05,
5.1931172555043525e-05,
5.4487373271889406e-05,
5.441295442908346e-05,
5.244336405529954e-05,
5.3145550435227854e-05,
5.456377880184332e-05,
5.353805427547363e-05,
5.4869452124935993e-05,
5.4083430619559654e-05,
5.5740460829493094e-05,
5.639764976958526e-05,
5.859853046594982e-05,
```

```
5.824338453661035e-05,
5.826919098822325e-05,
5.556401433691757e-05,
5.519995903737839e-05,
5.457114183307732e-05,
5.374794674859191e-05,
5.334013824884793e-05,
5.314544290834614e-05,
5.447293906810036e-05,
5.454684075780851e-05,
5.355013824884794e-05,
5.349856118791603e-05,
5.321536610343063e-05,
5.187121863799284e-05,
5.239533538146442e-05,
5.2020783410138254e-05,
5.162124423963134e-05,
5.109902201740912e-05,
5.0182309267793146e-05,
5.0475258576548906e-05,
5.0616912442396316e-05,
5.0254147465437793e-05,
4.895856118791603e-05,
4.878565796210958e-05,
5.057510496671787e-05,
5.138424475166411e-05,
5.020539170506913e-05,
4.901321556579621e-05,
4.90178853046595e-05,
4.9097557603686634e-05,
4.922224782386072e-05,
5.0083937532002046e-05,
5.071650281618023e-05,
5.041576548899129e-05,
5.11984536610343e-05,
5.182313364055299e-05,
5.159271889400922e-05,
4.901197132616488e-05,
4.6626589861751156e-05,
4.190917050691245e-05,
4.248188940092167e-05,
4.246015873015874e-05,
3.954983614951358e-05,
3.624764976958526e-05,
3.5417255504352284e-05,
3.4977235023041475e-05,
3.340626728110599e-05,
3.354810035842294e-05,
3.125009728622632e-05,
3.107963133640553e-05,
3.084239631336405e-05,
3.1276118791602663e-05,
2.9155873015873014e-05,
3.0289206349206345e-05,
3.1766195596518176e-05,
3.156967741935484e-05,
2.9974183307731693e-05,
```

```
2.9944690220174086e-05,
2.8283860727086532e-05,
2.8283860727086532e-05,
2.7183778801843315e-05,
2.926564260112647e-05,
2.752616487455197e-05,
2.738064516129032e-05,
2.732995391705069e-05,
2.7139308755760365e-05,
2.7346129032258062e-05,
2.7362309267793134e-05,
2.676764976958525e-05,
2.7530527393753197e-05,
2.7656733230926775e-05,
2.6632892985151046e-05,
2.8609262672811057e-05,
2.7984654377880182e-05,
2.826037378392217e-05,
3.0234290834613414e-05,
3.013584229390681e-05,
3.0037992831541217e-05,
3.0737516641065026e-05,
3.072271889400922e-05,
3.162990783410138e-05,
3.180797235023042e-05,
3.147834613415259e-05,
3.200283154121864e-05,
3.236467485919099e-05,
3.32683870967742e-05,
3.478392217101895e-05,
3.314644137224782e-05,
3.2352201740911416e-05,
3.2390481310803896e-05,
3.104178699436764e-05,
3.1034976958525344e-05,
3.139654889912955e-05,
3.1297235023041476e-05,
3.029375320020482e-05,
3.123801843317973e-05,
3.588087557603687e-05,
3.4255642601126475e-05,
3.376511520737327e-05,
3.4464551971326163e-05,
3.455479774705581e-05,
3.5520481310803894e-05,
3.552429083461342e-05,
3.680345622119816e-05,
3.59063031233999e-05,
3.704347670250896e-05,
3.9826932923707126e-05,
4.252142857142858e-05,
4.257780849974399e-05,
4.737377880184333e-05,
4.706963133640554e-05,
4.641903225806452e-05,
4.806275473630313e-05,
5.186091141833078e-05,
```

```
5.251275473630314e-05,
5.256802355350744e-05,
5.416955965181773e-05,
5.330533538146443e-05,
5.314579621095752e-05,
5.276865335381466e-05,
5.295544290834615e-05,
5.212382488479264e-05,
5.2488745519713274e-05,
5.256313364055301e-05,
5.0372032770097296e-05,
5.075684075780851e-05,
5.069232462877625e-05,
5.2347240143369185e-05,
5.248349206349207e-05,
5.2852539682539694e-05,
5.2798346134152595e-05,
5.286903225806453e-05,
5.1991648745519725e-05,
5.341121863799284e-05,
5.326080389144906e-05,
5.163091141833078e-05,
5.416866359447005e-05,
5.387848438300052e-05,
5.201009728622633e-05,
5.193894009216591e-05,
5.207877624167948e-05,
5.11878392217102e-05,
5.125339989759345e-05,
5.035189452124937e-05,
5.030273937532003e-05,
5.038400921658987e-05,
4.9873061955965194e-05,
4.96299436763953e-05,
4.9579406041986696e-05,
4.79694214029698e-05,
4.8083133640553004e-05,
4.83147209421403e-05,
4.82710189452125e-05,
5.005436763952893e-05,
5.013003584229391e-05,
4.921590373783922e-05,
4.953458269329237e-05,
5.3584147465437784e-05,
5.222930875576036e-05,
4.717747055811572e-05,
4.743225806451613e-05,
4.7381925243215566e-05,
4.613926267281106e-05,
4.5895442908346135e-05,
4.5380788530465955e-05,
4.516000000000001e-05,
4.4543614951356896e-05,
4.4994654377880194e-05,
4.314370711725551e-05,
4.538618023553508e-05,
4.1279170506912445e-05,
```

```
4.132543778801844e-05,
3.753461341525859e-05,
3.8119493087557614e-05,
3.8058315412186394e-05,
3.500240143369177e-05,
3.2242534562211994e-05,
3.117124423963135e-05,
3.2498545826932934e-05,
3.1137373271889415e-05,
3.098511520737328e-05,
3.295142857142858e-05,
3.348177163338454e-05,
3.3305714285714296e-05,
3.328687147977471e-05,
3.3157695852534564e-05,
3.426737327188941e-05,
3.4336948284690224e-05,
3.458456733230927e-05,
3.461929851510497e-05,
3.328521249359959e-05,
3.366635944700461e-05,
3.335207373271889e-05,
3.3027470558115714e-05,
3.303640552995392e-05,
3.4716082949308756e-05,
3.4162677931387604e-05,
3.418295442908346e-05,
3.3859989759344594e-05,
3.23316026625704e-05,
3.285723502304147e-05,
3.2720911418330766e-05,
3.30389759344598e-05,
3.345533538146441e-05,
3.3535949820788525e-05,
3.497306195596517e-05,
3.530275473630311e-05,
3.512456733230926e-05,
3.503995903737838e-05,
3.4913292370711716e-05,
3.5638986175115196e-05,
3.508896057347669e-05,
3.514747055811571e-05,
3.505555043522784e-05,
3.4857086533538136e-05,
3.541232462877623e-05,
3.4778591909882224e-05,
3.475063492063491e-05,
3.4782785458269315e-05,
3.432801843317971e-05,
3.0797695852534546e-05,
3.071207373271888e-05,
3.0733323092677916e-05,
3.118447516641064e-05,
3.094275473630311e-05,
3.210300051203275e-05,
3.199532002048129e-05,
3.131252432155656e-05,
```

```
3.1342324628776225e-05,
3.573769585253454e-05,
3.5564475166410634e-05,
3.480543778801842e-05,
3.0343108038914475e-05,
3.047522785458268e-05,
3.0636036866359435e-05,
3.0539836149513554e-05,
2.9644229390680993e-05,
2.8964147465437775e-05,
2.9161710189452112e-05,
2.9603778801843305e-05,
2.9799743983614936e-05,
2.8686728110599065e-05,
2.972837685611878e-05,
2.9717777777777767e-05,
2.770295442908345e-05,
2.7260967741935473e-05,
3.0473492063492052e-05,
3.3383292370711715e-05,
3.29647926267281e-05,
3.169543778801842e-05,
3.638396825396824e-05,
3.622575012800818e-05,
3.65227188940092e-05,
3.7041817716333834e-05,
3.644590373783921e-05,
3.634286226318483e-05,
3.9425427547363015e-05,
3.9872165898617496e-05,
4.20379723502304e-05,
4.20074551971326e-05,
4.456884792626726e-05,
4.463928315412185e-05,
4.9541356886840744e-05,
4.928661034306194e-05,
4.994725550435227e-05,
4.969106502816179e-05,
4.893225806451612e-05,
5.21269329237071e-05,
5.2617270865335364e-05,
5.25780184331797e-05,
6.225258064516126e-05,
6.242482846902199e-05,
6.387392217101893e-05,
6.329709677419353e-05,
6.507837685611877e-05,
6.537032258064514e-05,
6.523888376856116e-05,
6.58579928315412e-05,
6.685361495135686e-05,
6.57701382488479e-05,
7.514035842293905e-05,
7.508026625704043e-05,
7.612181771633383e-05,
7.666618023553506e-05,
7.63233230926779e-05,
```

```
7.712059907834099e-05,
7.565512032770095e-05,
7.598135688684073e-05,
7.678651817716332e-05,
7.975972862263184e-05,
8.005727086533538e-05,
8.018264720942139e-05,
7.491193548387095e-05,
8.142511520737325e-05,
8.310317972350228e-05,
8.410031233998973e-05,
8.959048131080386e-05,
9.049092165898615e-05,
9.038186379928312e-05,
9.067565796210954e-05,
9.363200204813105e-05,
9.350110599078337e-05,
9.314816180235531e-05,
9.388479262672807e-05,
9.369562724014333e-05,
9.168751664106499e-05,
9.176829493087555e-05,
9.188479262672808e-05,
9.205200204813105e-05,
8.974447516641061e-05,
8.755834101382485e-05,
8.75702304147465e-05,
9.127046082949304e-05,
8.79175934459805e-05,
8.701908346134148e-05,
8.689564260112643e-05,
8.636640552995388e-05,
8.612013824884788e-05,
8.661641065028157e-05,
8.366995903737835e-05,
8.307349206349201e-05,
7.964410138248844e-05,
7.848087557603681e-05,
7.605564260112642e-05,
7.59698156682027e-05,
7.034769585253451e-05,
7.014092677931382e-05,
7.16908499743983e-05,
7.221596518177158e-05,
7.203267793138756e-05,
6.824435227854576e-05,
6.68313261648745e-05,
6.68313261648745e-05,
5.852289298515099e-05,
5.8108530465949765e-05,
5.610041986687143e-05,
5.6366082949308706e-05,
5.476221198156677e-05,
5.467240143369171e-05,
5.502640552995387e-05,
5.546296979006651e-05,
5.42460727086533e-05,
```

```
5.4019774705581104e-05,
4.5749774705581104e-05,
4.597322580645156e-05,
4.544273937531997e-05,
4.4129237071172505e-05,
4.629353814644132e-05,
4.5438699436763906e-05,
4.570124423963129e-05,
4.555490527393748e-05,
4.3289344598054225e-05,
3.983253968253963e-05,
4.046349206349202e-05,
4.0073799283154074e-05,
4.019322580645157e-05,
3.53371326164874e-05,
3.2857255504352234e-05,
3.109253456221193e-05,
2.693963645673318e-05,
2.657601126472089e-05,
2.65629032258064e-05,
2.5925714285714235e-05,
2.415253456221193e-05,
2.557031233998971e-05,
2.4897808499743932e-05,
2.445747055811567e-05,
2.6108223246287713e-05,
2.6053179723502254e-05,
2.5810552995391654e-05,
2.7062739375319968e-05,
2.6568202764976908e-05,
2.6533384536610292e-05,
2.6552857142857094e-05,
2.6548484383000463e-05,
2.362525857654885e-05,
2.3071879160266207e-05,
2.3189774705581106e-05,
2.314336917562719e-05,
2.2850204813107988e-05,
2.2861141833077265e-05,
2.2230921658986126e-05,
2.2268699436763903e-05,
2.2400926779313824e-05,
2.6223937532001996e-05,
2.6223937532001996e-05,
2.7390527393753144e-05,
2.7391428571428518e-05,
2.740639528929846e-05,
2.8622677931387552e-05,
2.604654889912949e-05,
2.5381843317972295e-05,
2.5761679467485863e-05,
2.560755760368658e-05,
2.4016825396825336e-05,
2.3964838709677358e-05,
2.376811571940598e-05,
2.4406682027649705e-05,
2.443801843317966e-05,
```

```
2.4740849974398296e-05,
2.450479774705575e-05,
2.4034516129032197e-05,
2.4034516129032197e-05,
2.3861817716333784e-05,
2.341143369175621e-05,
2.4106932923707053e-05,
2.4176129032258e-05,
2.4413456221198093e-05,
2.410598054275467e-05,
2.4070803891448988e-05,
2.3086287762416728e-05,
2.2840266257040384e-05,
2.259820276497689e-05,
2.259415258576542e-05,
2.3217808499743915e-05,
2.3280389144905204e-05,
2.1816871479774635e-05,
2.1764352278545755e-05,
2.2069605734766953e-05,
2.1789344598054204e-05,
2.2418202764976885e-05,
2.268759344598047e-05,
2.363813108038907e-05,
2.5496195596518103e-05,
2.5502324628776166e-05,
2.5284792626728033e-05,
2.5143922171018868e-05,
2.434876088069629e-05,
2.481081925243208e-05,
2.479949820788523e-05,
2.3215842293906735e-05,
2.3193968253968177e-05,
2.3177327188940016e-05,
2.1466728110599002e-05,
2.2439631336405453e-05,
2.2684459805427472e-05,
2.3060322580645087e-05,
2.368522785458262e-05,
2.3666088069636382e-05,
2.3099493087557527e-05,
2.306209421402962e-05,
2.30841013824884e-05,
2.3102903225806377e-05,
2.3292109575012726e-05,
2.3275437788018355e-05,
4.242762416794667e-05,
4.268074244751656e-05,
3.918447004608287e-05,
4.172415258576541e-05,
4.25530414746543e-05,
4.7013870967741856e-05,
4.690926267281098e-05,
4.614166410650274e-05,
4.5942165898617436e-05,
4.6067327188940015e-05,
5.238476702508952e-05,
```

```
5.284842293906802e-05,
5.3161541218637914e-05,
5.330310803891441e-05,
5.526435227854575e-05,
5.50561955965181e-05,
5.949662570404498e-05,
5.9033722478238526e-05,
5.9024792626728035e-05,
5.900838709677412e-05,
5.9325151049667103e-05,
5.981977470558108e-05,
5.972834613415251e-05,
5.882521249359951e-05,
5.892167946748584e-05,
5.9703860727086455e-05,
5.96599078341013e-05,
5.980829493087549e-05,
5.8813824884792546e-05,
5.882598054275465e-05,
5.951506912442388e-05,
6.0013323092677845e-05,
5.945166410650273e-05,
6.05346543778801e-05,
6.0533292370711644e-05,
6.050794674859183e-05,
6.0214654377880106e-05,
6.118214029697893e-05,
6.065451612903218e-05,
6.0393686635944625e-05,
6.060759344598047e-05,
5.8436502816180166e-05,
5.847488991295436e-05,
6.013974398361488e-05,
5.976671786994361e-05,
6.018801843317966e-05,
5.9852534562211915e-05,
5.992321556579614e-05,
5.9943860727086466e-05,
6.0014244751664036e-05,
6.011869943676388e-05,
6.3439313876088e-05,
6.400800819252426e-05,
6.384628776241672e-05,
6.309859190988216e-05,
6.311994367639521e-05,
6.312805427547354e-05,
6.371805427547354e-05,
6.369235535074236e-05,
6.367293906810025e-05,
6.369644137224772e-05,
6.585295442908337e-05,
6.630596518177152e-05,
4.640691756272392e-05,
4.697645161290313e-05,
4.668289298515096e-05,
4.364952892985142e-05,
4.053641065028152e-05,
```

```
3.779119815668193e-05,
3.957020481310794e-05,
4.510304147465428e-05,
4.658644137224773e-05,
4.6532165898617415e-05,
4.150216589861742e-05,
4.386955965181762e-05,
4.3938561187915926e-05,
4.4035437788018335e-05,
4.2687972350230314e-05,
4.223379928315402e-05,
4.1434746543778703e-05,
4.172396825396815e-05,
4.3909999999999896e-05,
4.441930875576027e-05,
4.410977470558105e-05,
4.42216129032257e-05,
4.412627240143359e-05,
4.414146441372237e-05,
4.579866871479764e-05,
4.491166410650271e-05,
4.4908883768561084e-05,
4.4995473630312235e-05,
4.4996036866359345e-05,
4.836704045058873e-05,
4.789899129544281e-05,
5.178598054275463e-05,
5.174934459805417e-05,
5.0581141833077214e-05,
5.0598940092165794e-05,
5.068769585253446e-05,
5.0706226318484283e-05,
5.210837685611869e-05,
5.2101986687147874e-05,
5.899780849974388e-05,
5.775166410650271e-05,
6.70174091141832e-05,
7.044152073732709e-05,
7.048723502304138e-05,
7.207146441372239e-05,
7.12110343061955e-05,
7.155770097286217e-05,
8.188912954429074e-05,
8.730658986175107e-05,
8.805456733230919e-05,
9.181284690220167e-05,
0.0001003925089605734,
9.853175115207366e-05,
9.844788018433173e-05,
0.0001039824628776241,
0.00010643912954429077,
0.00010656290322580638,
0.00011590811571940597,
0.0001157335023041474,
0.00011582942140296973,
0.00011881866871479769,
0.00011693479262672805,
```

```
0.00011881313364055293,
0.00011964142857142851,
0.00012668630312339982,
0.00013481801843317966,
0.00013443339989759338,
0.0001370259651817716,
0.00013880788018433177,
0.00013904931387608802,
0.00013421687147977467,
0.00013638876088069632,
0.00013643888376856114,
0.0001377682027649769,
0.00013442522785458264,
0.00013485748591909877,
0.0001339360829493087,
0.00013443992370711725,
0.00014840691756272395,
0.00014627368663594465,
0.0001463035381464413,
0.00014472995903737834,
0.00014466603686635939,
0.00014436576036866352,
0.00014350575012800812,
0.00014390884792626722,
0.00014404515104966714,
0.00014498843830005115,
0.00014483519713261642,
0.00014530035842293901,
0.00014511529953917045,
0.00014724386072708647,
0.00014423157194060413,
0.00014481052739375312,
0.0001401579759344598047,
0.00014049078341013817,
0.00014073594470046075,
0.00014198110599078332,
0.00014276794674859185,
0.00014252442396313355,
0.0001402388868407578,
0.00014983880696364557,
0.00014346101894521242,
0.00014498218637992824,
0.00013566630312339983,
0.00013184741935483863,
0.00013342974398361488,
0.0001317203737839221,
0.00013261929851510489,
0.00013463175115207364,
0.00013463175115207364,
0.0001299530005120327,
0.00013029499743983608,
0.00012911283154121858,
0.00012911283154121858,
0.0001291167178699436,
0.00013299063492063485,
0.00013019124935995897,
0.00013765806451612896,
```

```
0.0001384277931387608,
0.00013299284690220165,
0.00013325856118791597,
0.00013905769585253447,
0.0001366231336405529,
0.00014032837685611 87,
0.00014220487455197123,
0.00014772515104966708,
0.00014139553507424465,
0.00014768564260112638,
0.00014740732718893998,
0.00014584759344598043,
0.00014447063492063482,
0.00014365023041474644,
0.00014664306195596506,
0.00014586313364055287,
0.00014566447004608283,
0.00014383626728110585,
0.00015470316948284677,
0.00015436264720942127,
0.0001543633640552994,
0.00015673087557603672,
0.00014669568868407563,
0.00014682413722478223,
0.00014646736303123382,
0.00014676214029697883,
0.00015140177163338437,
0.00015276779313876072,
0.00015232547363031216,
0.000154705151049667,
0.00015469876087 08806962,
0.00015205861751152057,
0.00015404210957501262,
0.00015373135688684058,
0.00016335216589861734,
0.00016061461341525843,
0.00016448988223246273,
0.00016517467485919083,
0.00016674049667178683,
0.0001689914490527392,
0.00017057780849974381,
0.0001783714285714284,
0.00017827400921658967,
0.00018863382488479245,
0.00018871216589861736,
0.0001814429032258063,
0.00018872488991295429,
0.00018970177163338438,
0.00019170091141833062,
0.00019752074244751646,
0.0001946986635944 6989,
0.00019403737839221693,
0.00019813661034306178,
0.00019959797235023023,
0.00019078112647209403,
0.00020437977470558097,
0.00020461175115207353,
```

```
 0.00020264912954429064,
 0.00019233422939068083,
 0.0001931749359959036,
 0.00019099220174091127,
 0.000191952857142857,
 0.00018096129032258 05,
 0.00018061705069124409,
 0.00017787350230414733,
 0.0001789682078853045,
 0.00017573478238607258,
 0.0001763049820788529,
 0.00017667945212493586,
 0.0001734000512032 7688,
 0.000166764792626728,
 0.00016514472094214018,
 0.00016082027649769575,
 0.00016081768561187904,
 0.0001598753353814643,
 0.0001606779467485918,
 0.00016698095750127998,
 0.0001636218177163337 4,
 0.0001627322119815667,
 0.00018583635944700448,
 0.0001847254121863798,
 0.00017701336917562711,
 0.0001815343317972349,
 0.00018421800819252418,
 0.0002098428110599077,
 0.0002161689759344597,
 0.00022357207373271878,
 0.0002291960368663 5934,
 ...],
1448)
```

In [162]:
```
var_e_q3_63 = np.zeros(shape=(1448,1))
for col_index in range(eqt_risk_prem_df.shape[1]):
    ri_minus_rf = eqt_risk_prem_df.iloc[:, col_index]
    rm_minus_rf = mkt_risk_prem_df[["Market Risk Premium"]]
    col_vars = []
    for i in range(1448):
        model = OLS(ri_minus_rf[i:i+62], add_constant(rm_minus_rf[i:i+62
]))
        res = model.fit()
        varis = res.resid
        col_vars.append(np.var(varis))
    var_e_q3_63 = np.c_[var_e_q3_63, col_vars]
var_e_q3_63
```

Out[162]:
```
array([[0.00000000e+00, 3.88278167e-04, 2.37273297e-04, ...,
        1.09215757e-03, 3.43703982e-04, 1.21146211e-03],
       [0.00000000e+00, 3.88152295e-04, 2.37854832e-04, ...,
        1.09327570e-03, 4.02094536e-04, 1.21369986e-03],
       [0.00000000e+00, 3.88324918e-04, 2.44060545e-04, ...,
        1.10035869e-03, 3.47754641e-04, 1.21060880e-03],
       ...,
       [0.00000000e+00, 8.87651773e-05, 3.92475094e-04, ...,
        1.01033499e-03, 2.89565134e-04, 1.13675361e-03],
       [0.00000000e+00, 8.87462678e-05, 3.96649166e-04, ...,
        1.04796946e-03, 2.89027862e-04, 1.13307292e-03],
       [0.00000000e+00, 8.87714491e-05, 3.95288783e-04, ...,
        1.04857770e-03, 2.82756791e-04, 9.51713160e-04]])
```

In [163]:
```
vars_df_q3_63 = pd.DataFrame(var_e_q3_63).drop(0, axis = 1)
vars_df_q3_63
```

Out[163]:

	1	2	3	4	5	6	7	8	9
0	0.000388	0.000237	0.000973	0.000625	0.000196	0.000434	0.001506	0.000115	0.000176
1	0.000388	0.000238	0.000956	0.000614	0.000199	0.000435	0.001449	0.000096	0.000178
2	0.000388	0.000244	0.000970	0.000616	0.000198	0.000428	0.001276	0.000087	0.000179
3	0.000378	0.000244	0.000919	0.000632	0.000203	0.000431	0.001189	0.000087	0.000186
4	0.000379	0.000252	0.000918	0.000634	0.000189	0.000431	0.001156	0.000086	0.000191
...	...	...	...	...	...	...	...	...	...
1443	0.000090	0.000399	0.001592	0.000142	0.000184	0.000189	0.000300	0.000104	0.001438
1444	0.000090	0.000392	0.001582	0.000144	0.000183	0.000189	0.000361	0.000104	0.001416
1445	0.000089	0.000392	0.001560	0.000144	0.000182	0.000202	0.000365	0.000104	0.001352
1446	0.000089	0.000397	0.001578	0.000145	0.000182	0.000204	0.000368	0.000103	0.001369
1447	0.000089	0.000395	0.001578	0.000147	0.000184	0.000206	0.000371	0.000098	0.001365

1448 rows × 100 columns

iii) Using the market capitalization weights (from the last day in the rolling window)of your securities, estimate the variance and standard deviation of your portfolio.

Formula:

$$\hat{\sigma}_p^2 = Var(\tilde{r}_p) = w^T \hat{\beta} \hat{\sigma}_M^2 \hat{\beta}^T w + w^T \hat{\Delta} w$$

In [164]:
```
df_weights_63 = df_weights.tail(1448)
df_weights_63
```

Out[164]:

Ticker #	1	2	3	4	5	6	7	8	9
62	0.019876	0.039362	0.001319	0.000330	0.003897	0.013208	0.000505	0.007949	0.000241
63	0.019960	0.040117	0.001373	0.000328	0.003792	0.013172	0.000508	0.007900	0.000238
64	0.019807	0.039649	0.001360	0.000336	0.003860	0.013410	0.000537	0.007826	0.000233
65	0.019982	0.040530	0.001374	0.000334	0.003871	0.013246	0.000521	0.007873	0.000229
66	0.019650	0.038668	0.001337	0.000330	0.003896	0.013063	0.000520	0.007886	0.000228
...	...	...	...	...	...	...	...	...	...
1505	0.013604	0.020282	0.000935	0.000440	0.005093	0.236809	0.000689	0.009864	0.000120
1506	0.013457	0.020469	0.000932	0.000440	0.005034	0.242061	0.000694	0.009756	0.000119
1507	0.013420	0.020067	0.000895	0.000433	0.004990	0.243796	0.000701	0.009732	0.000123
1508	0.013447	0.020065	0.000892	0.000441	0.004955	0.241941	0.000713	0.009759	0.000125
1509	0.013719	0.020300	0.000894	0.000437	0.004960	0.243645	0.000721	0.009688	0.000126

1448 rows × 100 columns

In [165]:
```
var_port_q3_63 = []
for i in range(len(df_weights_63)):
    diag_mat = np.diag(vars_df_q3_63.iloc[i])
    res = var_portfolio(df_weights_63.iloc[i], betas_df_q3_63.iloc[i], v
ar_rm_q3_63[i], diag_mat)
    var_port_q3_63.append(res)
arr_var_q3_63 = np.array(var_port_q3_63)
arr_var_q3_63
```

Out[165]:
```
array([7.63172963e-05, 8.01567192e-05, 8.07554042e-05, ...,
       1.09152319e-04, 1.08705594e-04, 1.09124494e-04])
```

In [166]:
```
arr_sd_q3_63 = np.sqrt(arr_var_q3_63)
arr_sd_q3_63
```

Out[166]:
```
array([0.00873598, 0.00895303, 0.0089864 , ..., 0.0104476 , 0.0104262 ,
       0.01044627])
```

Processing math: 100%

iv) Using the market capitalization weights and returns (from the day following the last day in the rolling window) of your securities, calculate the one-day ahead return of the portfolio, $\tilde{r}_p$.

In [260]:
```
#getting one-day ahead returns array
dayahead63_port_ret_q3 = []
dayahead63_ret_q3 = portfolio_q2_ret.loc[63:1510].to_numpy()
dayahead63_w_q3 = weights_q2_63.loc[63:1510].to_numpy()
for i in range(0, len(dayahead63_w_q2)):
    dayahead63_port_ret_q3.append(np.multiply(dayahead63_ret_q3[i], daya
head63_w_q3[i]))
dayahead63_port_ret_q3_arr = np.sum(np.array(dayahead63_port_ret_q3), ax
is = 1)
dayahead63_port_ret_q3_arr
```

Out[260]:
```
array([ 0.01740121,  0.0116785 , -0.00496981, ..., -0.00414611,
        0.00514476, -0.00946444])
```

v) Calculate the standardized outcome, $\tilde{z}_p$, where $\tilde{z}_p = \frac{\tilde{r}_p}{\hat{\sigma}_p}$ where we make the simplifying assumption that $E[\tilde{r}_p] = 0$.

In [168]:
```
standardized_outcomes_63_q3 = dayahead63_port_ret_q3_arr / arr_sd_q3_63
std_outcomes_63_q3 = pd.DataFrame(standardized_outcomes_63_q3)
std_outcomes_63_q3.index += 63
std_outcomes_63_q3.rename(columns={0: "Standardized Outcome"}, inplace =
True)
std_outcomes_63_q3
```

Out[168]:

	Standardized Outcome
63	1.991902
64	1.304419
65	-0.553037
66	-0.527825
67	0.198191
...	...
1506	1.133099
1507	0.485674
1508	-0.396848
1509	0.493445
1510	-0.906012

1448 rows × 1 columns

b) Compute bias statistics.

In [169]:
```
bias_stat_q3_504 = np.std(standardized_outcomes_504_q3)
bias_stat_q3_252 = np.std(standardized_outcomes_252_q3)
bias_stat_q3_126 = np.std(standardized_outcomes_126_q3)
bias_stat_q3_63 = np.std(standardized_outcomes_63_q3)
print(bias_stat_q3_504, bias_stat_q3_252, bias_stat_q3_126, bias_stat_q3
_63)
```

```
1.3118478769931892 1.1320711225951727 1.0749500838326524 1.057843577865
541
```

Question 5

In [239]:
```
#randomized portfolio indices
stocks = pd.read_csv("fifty_portfolios.csv")
stocks = stocks[0:50]
stocks.head()
```

Out[239]:

	984	1236	505	1235	1552	1732	918	863	169	1157	...	349	1588	1731	616	5
0	1757	1199	1673	1702	1511	86	1428	307	170	159	...	1044	238	1060	1646	13
1	1212	1755	1549	1121	193	1270	679	34	1816	1626	...	1247	484	141	32	5
2	425	1082	1616	784	1197	1067	215	410	1752	1081	...	1785	1797	341	997	2
3	1006	110	980	273	682	1473	887	289	989	1341	...	508	792	2	1356	13
4	26	1386	766	1435	466	1792	1865	1491	17	79	...	1577	1685	1088	247	

5 rows × 100 columns

In [240]:
```
best_rolling_window_q3 = 63
print("Best Rolling Window Market Model:", 63)
```

```
Best Rolling Window Market Model: 63
```

In [241]:
```
betas_df_q3_63
```

Out[241]:

	1	2	3	4	5	6	7	8	9
0	2.052812	1.631485	2.467977	0.149854	0.834601	1.489859	2.200903	0.891679	1.255516
1	2.024280	1.610569	2.470555	0.134584	0.813522	1.469599	2.103833	0.854639	1.232819
2	2.029893	1.617511	2.626662	0.150088	0.836575	1.398232	2.441186	0.785010	1.221262
3	2.002958	1.605841	2.581234	0.202040	0.870995	1.430943	2.490066	0.782768	1.185146
4	1.986856	1.585203	2.553937	0.221884	0.851413	1.438855	2.489432	0.773026	1.202064
...	...	...	...	...	...	...	...	...	...
1443	1.155123	1.901754	1.162571	0.910696	0.745226	0.835353	1.144480	0.652799	1.932211
1444	1.152748	1.883088	1.184261	0.903087	0.751742	0.834111	1.189235	0.650191	1.972411
1445	1.172412	1.908703	1.270773	0.916972	0.768153	0.839263	1.201479	0.652601	2.109775
1446	1.172805	1.906460	1.270128	0.914495	0.766944	0.837897	1.198084	0.654865	2.110638
1447	1.173619	1.902094	1.272323	0.905755	0.772203	0.841758	1.190653	0.665431	2.118114

1448 rows × 100 columns

In [242]:
```
df_weightz = secdata_cap_group.iloc[:, :].apply(lambda x: x.div(x.sum
()), axis=1)
df_weightz
```

Out[242]:

Ticker #	1	2	3	4	5	6	7	8	9
0	0.001326	0.003143	0.000100	0.000023	0.000291	0.000759	0.000031	0.000586	0.000015
1	0.001345	0.003207	0.000099	0.000023	0.000285	0.000781	0.000029	0.000598	0.000015
2	0.001385	0.003178	0.000105	0.000023	0.000289	0.000777	0.000032	0.000595	0.000016
3	0.001399	0.003145	0.000108	0.000023	0.000297	0.000793	0.000033	0.000599	0.000016
4	0.001456	0.003169	0.000100	0.000023	0.000288	0.000816	0.000034	0.000594	0.000016
...	...	...	...	...	...	...	...	...	...
1506	0.000934	0.001421	0.000065	0.000031	0.000350	0.016806	0.000048	0.000677	0.000008
1507	0.000936	0.001399	0.000062	0.000030	0.000348	0.016996	0.000049	0.000678	0.000009
1508	0.000935	0.001395	0.000062	0.000031	0.000345	0.016821	0.000050	0.000679	0.000009
1509	0.000959	0.001419	0.000062	0.000031	0.000347	0.017025	0.000050	0.000677	0.000009
1510	0.000977	0.001415	0.000063	0.000030	0.000345	0.017102	0.000051	0.000669	0.000009

1511 rows × 1877 columns

In [243]:
```
def market_beta(w, b):
    return np.dot(np.transpose(w), b)
```

a)

Date: 12/30/2005 (idx loc = 503)

In [256]:
```
betas63_q3_2005 = []
for i in range(50):
    beta_m = betas_df_q3_63.iloc[63]
    weights = df_weightz[list(stocks.iloc[i])].iloc[63]
    betas_market = market_beta(weights, beta_m)
    betas63_q3_2005.append(betas_market)
betas63_q3_2005
```

Out[256]:
```
[0.052476585476291554,
 0.060517796382025806,
 0.05266254554281719,
 0.04484296288982689,
 0.05008550819633388,
 0.11483905345383778,
 0.073322857947415,
 0.05505125994248011,
 0.12427471755206039,
 0.1163201164825795,
 0.0708626674840099,
 0.05541155423131752,
 0.05432297537972558,
 0.13442441034486619,
 0.10585230959008153,
 0.03577555334839695,
 0.09465118512066485,
 0.0714363304019478,
 0.03363909212684032,
 0.04195793292668462,
 0.04117789319176979,
 0.07031441366385044,
 0.05381480562401819,
 0.078138296288351,
 0.10093688542026702,
 0.051036879516628914,
 0.07667610383582535,
 0.06238527748300194,
 0.04332837367609722,
 0.03933712335535698,
 0.0724217109221468,
 0.061762543729987865,
 0.06345742579112622,
 0.07193427218599058,
 0.0694555719928294,
 0.07449283302234419,
 0.05083367356123818,
 0.06286017502275587,
 0.04995425185709235,
 0.047783139703163544,
 0.12594856391676224,
 0.06441863345230071,
 0.035499326887140074,
 0.04869310649040237,
 0.03921184699157464,
 0.04776171596102408,
 0.04421140014000223,
 0.09845489794086111,
 0.09824484835190558,
 0.042846074038160334]
```

Date: 12/31/2007 (idx loc = 1005)

Processing math: 100%

In [255]:
```
betas63_q3_2007 = []
for i in range(50):
    beta_m = betas_df_q3_63.iloc[942]
    weights = df_weightz[list(stocks.iloc[i])].iloc[942]
    betas_market = market_beta(weights, beta_m)
    betas63_q3_2007.append(betas_market)
betas63_q3_2007
```

Out[255]:
```
[0.04844186126740678,
 0.05022385833208992,
 0.04548861529658407,
 0.05461929366595343,
 0.061548261018475034,
 0.06534954195811013,
 0.05688790637873255,
 0.046104948881788836,
 0.1076617144065238,
 0.11378194636284328,
 0.06277894880416302,
 0.05626732098236807,
 0.06218068742784019,
 0.13614324748653606,
 0.06585317014217026,
 0.03614911172060497,
 0.08520697207263117,
 0.06860030394451724,
 0.029300436262620907,
 0.045219318082515836,
 0.04961497148371338,
 0.05606900886658664,
 0.046407368801384026,
 0.06279147012171692,
 0.07378251846715525,
 0.06136129977648882,
 0.05525272969212655,
 0.05152475209489002,
 0.051416253736796196,
 0.03044853664343607,
 0.05956111460030072,
 0.05658907816646388,
 0.047019011027107045,
 0.05592551905126991,
 0.08339418247962005,
 0.07113678084787348,
 0.045472694165700295,
 0.06119284379865997,
 0.05623640029741284,
 0.04611064931309587,
 0.0688250843181559,
 0.06321828661968756,
 0.04204620124916704,
 0.05544096881555913,
 0.04857027940311408,
 0.05179883276458884,
 0.0585866463622222,
 0.09813033073045757,
 0.06553974079301855,
 0.052732289980279914]
```

Date: 12/31/2009 (idx loc = 1510)

Processing math: 100%

In [258]:
```
betas63_q3_2009 = []
for i in range(50):
    beta_m = betas_df_q3_63.iloc[1447]
    weights = df_weightz[list(stocks.iloc[i])].iloc[1447]
    betas_market = market_beta(weights, beta_m)
    betas63_q3_2009.append(betas_market)
betas63_q3_2009
```

Out[258]:
```
[0.041969788001929145,
 0.03609766003979558,
 0.0571643240845565,
 0.05099314794069944,
 0.06269323967843796,
 0.07007908597911089,
 0.06542687007033943,
 0.06294522474036876,
 0.1326222012094358,
 0.09869831219725504,
 0.08662925378641319,
 0.05210428234069958,
 0.07351577229722664,
 0.13157672809365403,
 0.06979837215389698,
 0.042045514913615294,
 0.07813386136379571,
 0.08461502709588789,
 0.02988254295835566,
 0.04835438020553477,
 0.05411589350054576,
 0.06816890698885693,
 0.045398859636565696,
 0.05664138297940627,
 0.0667691865821991,
 0.0686110393577253,
 0.06398265892236549,
 0.054928708373972485,
 0.06452772300637954,
 0.03939297329802155,
 0.06403339955640502,
 0.04670969799885384,
 0.054133111071517415,
 0.06748490015918884,
 0.08095751710068065,
 0.0926428927302992,
 0.04914394122850718,
 0.05888155868746275,
 0.07462290285725254,
 0.050435009620435664,
 0.08475775674172936,
 0.08500303631759625,
 0.04219611133616479,
 0.0776177220368038,
 0.05513947908324862,
 0.04570813188434042,
 0.059986261836244156,
 0.11240965572216283,
 0.08808594585756582,
 0.06886014406080077]
```

b)

Processing math: 100%

Date: 12/30/2005 (idx loc = 503)

In [293]:
```
#excludes the first 63 days
rp_2005 = 1/63*np.sum((dayahead63_port_ret_q3_arr[378:441] - np.array(ff
data['Risk-free rate'])[441:504]))
rp_2005
```

Out[293]: 0.0010859704382670666

In [290]:
```
rm_2005 = 1/63*np.sum((np.array(ffdata['Market Returns'])[441:504] - np.
array(ffdata['Risk-free rate'])[441:504]))
rm_2005
```

Out[290]: 8.365079365079375e-05

Date: 12/31/2007 (idx loc = 1005)

In [286]:
```
rp_2007 = 1/63*np.sum((dayahead63_port_ret_q3_arr[880:943] - np.array(ff
data['Risk-free rate'])[943:1006]))
rp_2007
```

Out[286]: -0.00011254414214054921

In [291]:
```
rm_2007 = 1/63*np.sum((np.array(ffdata['Market Returns'])[943:1006] - np
.array(ffdata['Risk-free rate'])[943:1006]))
rm_2007
```

Out[291]: -0.0008863492063492066

Date: 12/31/2009 (idx loc = 1510)

In [289]:
```
rp_2009 = 1/63*np.sum((dayahead63_port_ret_q3_arr[1385:1448] - np.array(
ffdata['Risk-free rate'])[1448:1511]))
rp_2009
```

Out[289]: 0.001998517909506608

In [292]:
```
rm_2009 = 1/63*np.sum((np.array(ffdata['Market Returns'])[1448:1511] - n
p.array(ffdata['Risk-free rate'])[1448:1511]))
rm_2009
```

Out[292]: 0.0013793650793650792

In [ ]:
```

```