additional information regarding MC1-Project-2

Here is some clarification in response to a number of questions on piazza about Project-2:

You should assume that the following functions will be available to you, implemented by us in analysis.py:

get\_portfolio\_value(prices, allocs, start\_val)

get\_portfolio\_stats(port\_val, daily\_rf, samples\_per\_year)

You will need these in order to assess the performance of the portfolio with the specified allocations of the current portfolio.  You can also, if you like implement this functionality directy in your optimization.py code.

**From Piazza:**

You should assume the risk free rate is zero.

 all of our test cases will have 4 stocks.

# MC1-Project-2

## Contents

 [[hide](http://quantsoftware.gatech.edu/MC1-Project-2)]

* [1 Overview](http://quantsoftware.gatech.edu/MC1-Project-2#Overview)
* [2 Task](http://quantsoftware.gatech.edu/MC1-Project-2#Task)
* [3 Template](http://quantsoftware.gatech.edu/MC1-Project-2#Template)
* [4 Suggestions](http://quantsoftware.gatech.edu/MC1-Project-2#Suggestions)
* [5 Example output 1](http://quantsoftware.gatech.edu/MC1-Project-2#Example_output_1)
* [6 Example output 2](http://quantsoftware.gatech.edu/MC1-Project-2#Example_output_2)
* [7 Example output 3](http://quantsoftware.gatech.edu/MC1-Project-2#Example_output_3)
* [8 Example output 4](http://quantsoftware.gatech.edu/MC1-Project-2#Example_output_4)
* [9 What to turn in](http://quantsoftware.gatech.edu/MC1-Project-2#What_to_turn_in)
* [10 Rubric](http://quantsoftware.gatech.edu/MC1-Project-2#Rubric)

## Overview

In this project you will use what you learned about optimizers to optimize a portfolio. That means that you will find how much of a portfolio's funds should be allocated to each stock so as to optimize it's performance. In this case we define "optimal" as maximum Sharpe ratio.

You will leverage the functions you created in the last project that assessed the value of a portfolio with a given set of allocations.

## Task

Implement a Python function named find\_optimal\_allocations() that can find the optimal allocations for a given stock portfolio. You should optimize for Sharpe ratio.

The function should accept as input historical stock prices (supplied as a pandas dataframe, with each column representing one equity), and return a list of floats (as a one-dimensional numpy array) that represents the allocations to each of the equities. Use functions developed in the portfolio analysis project to compute daily portfolio value and statistics.

## Template

Instructions:

* Implement the find\_optimal\_allocations() function in mc1\_p2/portfolio/optimization.py.
* To execute, run **python -m portfolio.optimization** from mc1\_p2/ directory.

A helper function (optimize\_portfolio) has been included in the template code (see mc1\_p2/portfolio/optimization.py), which is called with the desired date range and symbols (see test\_run()):

start\_date = '2010-01-01'

end\_date = '2010-12-31'

symbols = ['GOOG', 'AAPL', 'GLD', 'XOM']

**optimize\_portfolio**(start\_date, end\_date, symbols)

This in turn reads stock price data, and calls the function to find optimal allocations (you need to implement this):

allocs = **find\_optimal\_allocations**(prices)

Your solution to the optimization problem should leverage the functions you wrote in the last assignment, namely

* **get\_portfolio\_value**(prices, allocs, start\_val): Compute daily portfolio value given stock prices, allocations and starting value.  
  Ensure that it returns a pandas Series or DataFrame (with a single column).
* **get\_portfolio\_stats**(port\_val, daily\_rf, samples\_per\_year): Calculate statistics on daily portfolio value, given daily risk-free rate and data sampling frequency.  
  This function should return a *tuple* consisting of the following statistics (in order): cumulative return, average daily return, standard deviation of daily return, Sharpe ratio  
  Note: The return statement provided ensures this order.
* **plot\_normalized\_data**(df, title, xlabel, ylabel): Normalize given stock prices and plot for comparison.  
  This is used to create a chart that illustrates the value of your portfolio over the year and compares it to SPY.  
  Note: Before plotting, portfolio and SPY values should be normalized to 1.0 at the beginning of the period. Also, use the plot\_data() utility function to generate and show your plot.

We will provide implementations of these functions that you can call from your optimizer code. Note that when your code is submitted for this project, you will not be submitting your analysis.py code, but instead your code will be calling our implementation of that. Accordingly, be sure you've implemented the API to assess\_portfolio() correctly!

The helper function (optimize\_portfolio()) prints out the optimal allocations, and some statistics. It also plots a chart comparing the daily value of your portfolio over the year and compares it to SPY. The portfolio and SPY are normalized to 1.0 at the beginning of the period. We will not be calling optimize\_portfolio to test your code, we will be calling find\_optimal\_allocations().

Make sure the output and plot look like the examples shown below.

## Suggestions

Refer to comments in find\_optimal\_allocations() for pointers regarding how to implement it. In order to specify bounds and constraints when using the scipy.optmizemodule, you'll need to use a special syntax explained here: <http://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html>

For bounds, you simply need to pass in a sequence of 2-tuples (<low>, <high>). Just remember that you need to supply as many tuples as the number of stocks in your portfolio.

For constraints, it's a little tricky. You need to pass in a sequence of dicts (dictionaries), one dictionary per constraint. Each dictionary must specify the type of constraint ('eq' for equality, or 'ineq' for inequality), and a function that *returns 0 only when the input satisfies the constraint* (this is the same input that is supplied to your evaluation function). E.g. to constrain the sum of all values in the input array to be less than 50, you could pass in the following (lambdas are just anonymous functions defined on-the-spot):

constraints = ({ 'type': 'ineq', 'fun': lambda inputs: 50.0 - np.sum(inputs) })

## Example output 1

Here's an example output for your function. These are actual correct values that you can use to check your work.

Start Date: 2010-01-01

End Date: 2010-12-31

Symbols: ['GOOG', 'AAPL', 'GLD', 'XOM']

Optimal allocations: [ 5.38105153e-16 3.96661695e-01 6.03338305e-01 -5.42000166e-17]

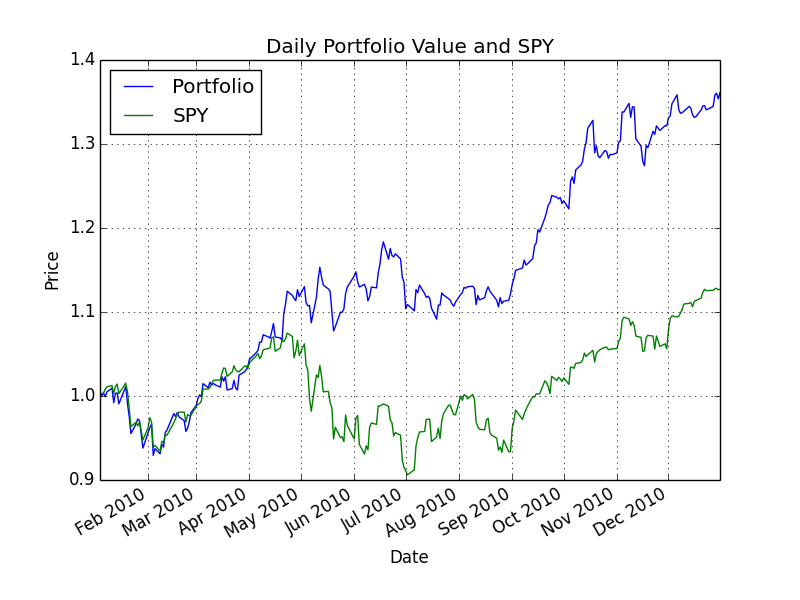
Sharpe Ratio: 2.00401501102

Volatility (stdev of daily returns): 0.0101163831312

Average Daily Return: 0.00127710312803

Cumulative Return: 0.360090826885

Corresponding comparison plot:

[](http://quantsoftware.gatech.edu/File:Comparison_optimal.png)

## Example output 2

Start Date: 2004-01-01

End Date: 2006-01-01

Symbols: ['AXP', 'HPQ', 'IBM', 'HNZ']

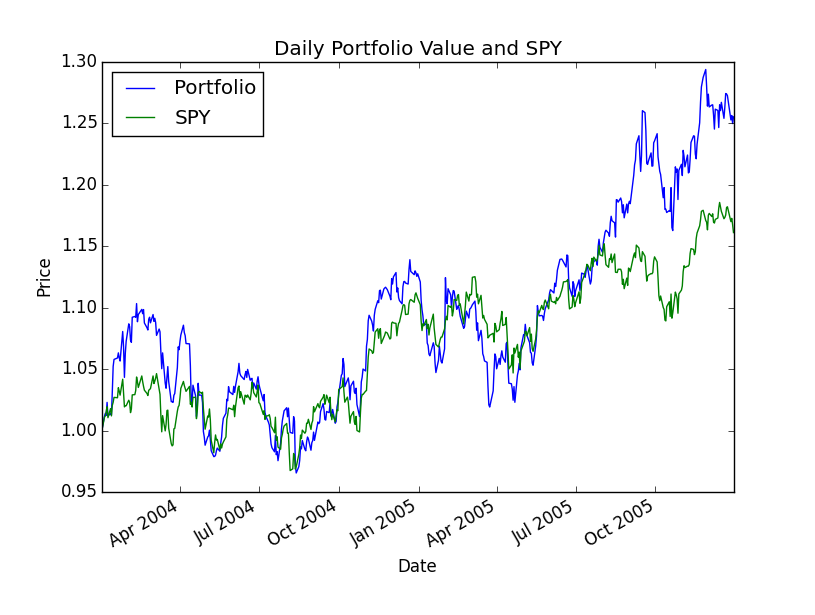
Optimal allocations: [ 7.75113042e-01 2.24886958e-01 -1.18394877e-16 -7.75204553e-17]

Sharpe Ratio: 0.842697383626

Volatility (stdev of daily returns): 0.0093236393828

Average Daily Return: 0.000494944887734

Cumulative Return: 0.255021425162

[](http://quantsoftware.gatech.edu/File:Example2.png)

## Example output 3

Start Date: 2004-12-01

End Date: 2006-05-31

Symbols: ['YHOO', 'XOM', 'GLD', 'HNZ']

Optimal allocations: [ -3.84053467e-17 7.52817663e-02 5.85249656e-01 3.39468578e-01]

Sharpe Ratio: 1.5178365773

Volatility (stdev of daily returns): 0.00797126844855

Average Daily Return: 0.000762170576913

Cumulative Return: 0.315973959221

## Example output 4

Start Date: 2005-12-01

End Date: 2006-05-31

Symbols: ['YHOO', 'HPQ', 'GLD', 'HNZ']

Optimal allocations: [ -1.67414005e-15 1.01227499e-01 2.46926722e-01 6.51845779e-01]

Sharpe Ratio: 3.2334265871

Volatility (stdev of daily returns): 0.00842416845541

Average Daily Return: 0.00171589132005

Cumulative Return: 0.229471589743

Minor differences in float values may arise due to different implementations.

## What to turn in

Be sure to follow these instructions diligently!

Via T-Square, submit as attachments (no zip files; refer to schedule for deadline):

* Your code as optimization.py (only the function find\_optimal\_allocations() will be tested)
* Plot comparing the optimal portfolio with SPY as comparison\_optimal.png using the following parameters:  
  Start Date: 2010-01-01, End Date: 2010-12-31, Symbols: ['GOOG', 'AAPL', 'GLD', 'HNZ']

Unlimited resubmissions are allowed up to the deadline for the project.

## Rubric

* Part 1: Chart is correct
  + Normalized values start at 1.0 on left (10%)
  + Shape of curves are correct (10%)
* Part 2: 10 test cases: We will test your code against 10 cases (8% per case). Each case will be deemed "correct" if:
  + sum(allocations) = 1.0 +- 0.02
  + Each allocation is between 0 and 1.0 +- 0.02 (negative allocations are allowed if they are very small)
  + Each allocation matches reference solution +- 0.10