Summer 2016 Project 1

From Quantitative Analysis Software Courses

Contents

Overview

A portfolio is a collection of stocks (or other assets) and corresponding allocations of funds to each of them. In order to evaluate and compare different portfolios, we first need to compute certain metrics, based on available historical data.

The primary goal of this assignment is to introduce you to this form of portfolio analysis. You will use pandas for reading in data, calculating various statistics and plotting a comparison graph.

Task

Create a function called `assess_portfolio()` that takes as input a description of a portfolio and computes important statistics about it.

You are given the following inputs for analyzing a portfolio:

- A date range to select the historical data to use (specified by a start and end date). You should consider performance from close of the start date to close of the end date.
- Symbols for equities (e.g., GOOG, AAPL, GLD, XOM). Note: You should support any symbol in the data directory.
- Allocations to the equities at the beginning of the simulation (e.g., 0.2, 0.3, 0.4, 0.1), should sum to 1.0.

- Total starting value of the portfolio (e.g. $1,000,000)

Your goal is to compute the daily portfolio value over given date range, and then the following statistics for the overall portfolio:

- Cumulative return
- Average period return (if sampling frequency == 252 then this is average daily return)
- Standard deviation of daily returns
- Sharpe ratio of the overall portfolio, given daily risk free rate (usually 0), and yearly sampling frequency (usually 252, the no. of trading days in a year)
- Ending value of the portfolio

Your program should include a helper function to specify the portfolio data, then your function should calculate and return the portfolio statistics. Be sure to include all necessary code in your single submitted Python file.

API specification

For grading purposes, we will test ONLY `assess_portfolio()` the function that computes statistics. You should implement the following API EXACTLY, if you do not, your submission will be penalized at least 25%.

```
import datetime as dt
cr, adr, sddr, sr, ev = \
    assess_portfolio(sd=dt.datetime(2008,1,1), ed=dt.datetime(2009,1,1), \
    syms=['GOOG','AAPL','GLD','XOM'], \
    allocs=[0.1,0.2,0.3,0.4], \
    sv=1000000, rfr=0.0, sf=252.0, \
    gen_plot=False)
```

Where the returned outputs are:

- cr: Cumulative return
- adr: Average period return (if sf == 252 this is daily return)
- sddr: Standard deviation of daily return
- sr: Sharpe ratio
- ev: End value of portfolio

The input parameters are:

- sd: A datetime object that represents the start date
- ed: A datetime object that represents the end date
- syms: A list of symbols that make up the portfolio (note that your code should support any symbol in the data directory)
- allocs: A list of allocations to the stocks, must sum to 1.0
- sv: Start value of the portfolio
- rfr: The risk free return per sample period for the entire date range. We assume that it does not change.
- sf: Sampling frequency per year

- gen_plot: If True, create a plot named plot.png

Template

A template is provided for you to get started with the project: mc1_p1.zip

Download and unzip it inside ml4t/. After you unzip it you should see the following directory structure:

- ml4t/: Root directory for course
  - data/: Location of data that you installed earlier
  - mc1_p1/: Root directory for this project
    - analysis.py: Main project script with functions you need to implement, as well as test code
    - output/: Directory to store all program outputs, including plots
    - util.py: Utility functions (do not modify these, unless instructed)

You should change ONLY analysis.py. ALL of your code should be in that one file. Do not create additional files. It should always remain in and run from the directory ml4t/mc1_p1/. If you move it somewhere else and develop your code there, it may not run properly when auto graded.

Notes:

- Ignore any file named __init__.py; they are used to mark directories as Python packages.
- We assume your data/ directory is one level up, (i.e., ../data/). That directory should contain all stock data, in CSV files (e.g. GOOG.csv, AAPL.csv, etc.)
- To execute the main script, make sure your current working directory is mc1_p1/, then run:

```
python analysis.py
```

Suggestions

Here is a suggested high-level outline for what your code needs to do:

- Read in adjusted closing prices for the 4 equities.
- Normalize the prices according to the first day. The first row for each stock should have a value of 1.0 at this point.
- Multiply each column by the allocation to the corresponding equity.
- Multiply these normalized allocations by starting value of overall portfolio, to get position values.
- Sum each row (i.e. all position values for each day). That is your daily portfolio value.
- Compute statistics from the total portfolio value.

In future projects, you will also want to compute portfolio statistics. To make that task easier, we suggest that you create a helper function within your code that has the following prototype. Note that we will not be testing this part of your code directly, so it isn't essential that you follow the API exactly.

```
cr, adr, sddr, sr = \
    compute_portfolio_stats(prices = df_prices, \
    allocs=[0.1,0.2,0.3,0.4],\
    rfr = 0.0, sf = 252.0)
```

Where the returned outputs are:

- cr: Cumulative return
- adr: Average daily return
- sddr: Standard deviation of daily return
- sr: Sharpe Ratio

The input parameters are:

- prices is a data frame or an ndarray of historical prices.
- allocs: A list of allocations to the stocks, must sum to 1.0
- rfr: The risk free return per sample period for the entire date range. We assume that it does not change.
- sf: Sampling frequency per year

Here are some notes and assumptions:

- When we compute statistics on the portfolio value, we do not include the first day.
- We assume you are using the data provided. If you use other data your results may turn out different from ours. Yahoo's online data changes every day. We cannot not build a consistent "correct" answer based on "live" Yahoo data.
- Assume 252 trading days/year.

Make sure your `assess_portfolio()` function gives correct output. Check it against the examples below.

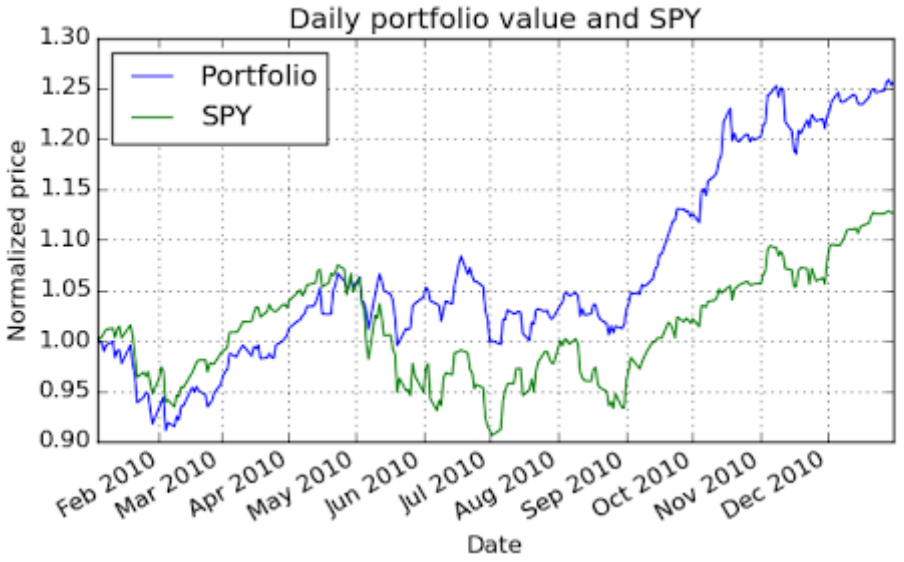
Example output

These are actual correct examples that you can use to check your work.

Example 1

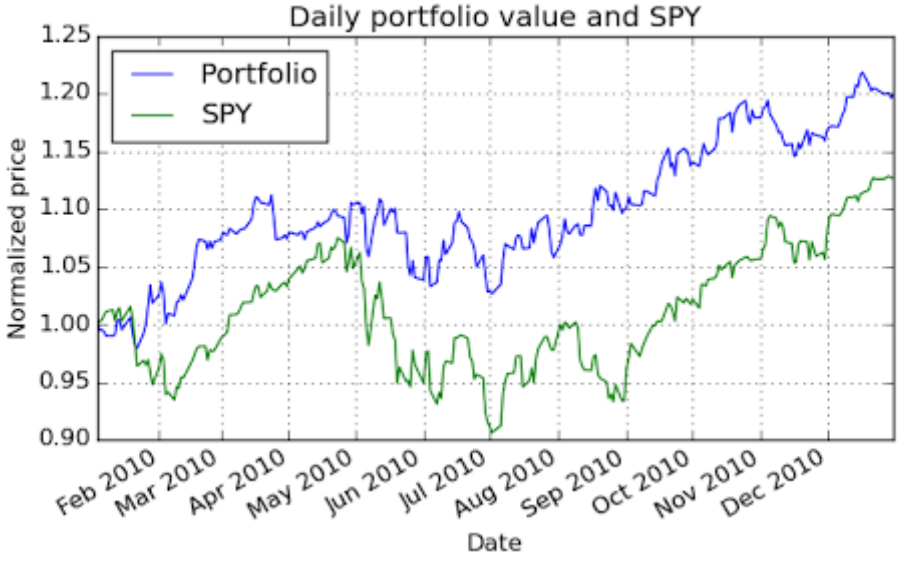
```
Start Date: 2010-01-01
End Date: 2010-12-31
Symbols: ['GOOG', 'AAPL', 'GLD', 'XOM']
Allocations: [0.2, 0.3, 0.4, 0.1]
Sharpe Ratio: 1.51819243641
Volatility (stdev of daily returns): 0.0100104028
```

```
Average Daily Return: 0.000957366234238
Cumulative Return: 0.255646784534
```



Example 2

```
Start Date: 2010-01-01
End Date: 2010-12-31
Symbols: ['AXP', 'HPQ', 'IBM', 'HNZ']
Allocations: [0.0, 0.0, 0.0, 1.0]
Sharpe Ratio: 1.30798398744
Volatility (stdev of daily returns): 0.00926153128768
Average Daily Return: 0.000763106152672
Cumulative Return: 0.198105963655
```



Example 3

```
Start Date: 2010-06-01
End Date: 2010-12-31
Symbols: ['GOOG', 'AAPL', 'GLD', 'XOM']
Allocations: [0.2, 0.3, 0.4, 0.1]
Sharpe Ratio: 2.21259766672
Volatility (stdev of daily returns): 0.00929734619707
Average Daily Return: 0.00129586924366
Cumulative Return: 0.205113938792
```

What to turn in

Be sure to follow these instructions diligently!

Submit via T-Square as attachments only (no zip files please):

- Your code as `analysis.py` (please use this EXACT filename)
- Your plot of daily portfolio value versus SPY as `plot.png` (use EXACT filename)
  The plot should be generated using the following: Start Date: 2009-01-01, End Date: 2009-12-31, Symbols: ['GOOG', 'AAPL', 'GLD', 'MSFT'], Allocations: [0.3, 0.3, 0.1, 0.3]

Important: We will test against OTHER symbols and other allocations, so don't hardcode the list of symbols.

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

Rubric

Part 1: Chart is correct (20 points)

- -20 no chart or chart is total nonsense
- Only one of these two:
  - -10 chart wrong shape (incl. wrong time period, etc)
  - -5 chart partly wrong shape (diverges from correct halfway through, etc)
- Only one of these two:
  - -10 chart not normalized
  - -5 chart data scale substantially wrong (right shape, normalized to 1, but max/end values wrong, 3.5 instead of 2.5 etc)
- -10 missing required data series (didn't plot one of portfolio or SPY)
- -2 chart labels/text unreadable (too small, labels overlap a lot, etc)
- Min score: 0.

Part 2: 10 test cases: We will test your code against 10 cases (8% per case). Each case will be deemed "correct" if:

- 4%: Sharpe ratio = reference answer +- 0.001 (4%)
- 2%: Average daily return = reference answer +- 0.00001
- 2%: Cumulative return = reference answer +- 0.001

Test Cases

The test cases used for grading this assignment in Spring 2016 are available below for you to further test your code. We will test your code with different test cases.

- MC1-Project-1-Test-Cases-spr2016

Required, Allowed & Prohibited

Required:

- Your project must be coded in Python 2.7.x.
- Your code must run on one of the university-provided computers (e.g. buffet02.cc.gatech.edu), or on one of the provided virtual images.
- Use the code for reading in historical data provided in util.py
- Your code must run in less than 5 seconds on one of the university-provided computers.

Allowed:

- You can develop your code on your personal machine, but it must also run successfully on one of the university provided machines or virtual images.
- Your code may use standard Python libraries (except os).
- You may use the NumPy, SciPy, matplotlib and Pandas libraries. Be sure you are using the correct versions.
- You may use code provided by the instructor, or allowed by the instructor to be shared.

Prohibited:

- Any use of global variables.
- Any libraries not listed in the "allowed" section above.
- Use of any code other than util.py to read in data.
- Use of Python's os module.
- Any code you did not write yourself.
