Homework 1
ORIE 5370, Spring 2017
Due noon Friday, February 3.

- Submit text files of your matlab programs to our TA, Matthew Zalesak (mdz32).
- In the subject line, list the Cornell netid's for all of your team's members.

Each group must write its own MATLAB programs. You are free to have discussions with members from other groups, but you cannot share code.

The homework in the course consists mostly of MATLAB problems meant to give you tools – and ways of thinking – for creating and doing projects. (You don't necessarily have to use these tools in your projects.) Homework – worth only 20% of your grade – is marked leniently. In particular, if your MATLAB program for a homework problem executes and gives the right answer, the grade for the problem will be an A, regardless of whether your code is elegant or looks a mess. (Elegant code, on the other hand, definitely can increase the grade on a project, unlike homework.) If your MATLAB program for a homework problem *does not* execute or give the right answer, then it will be examined more closely, and given a lower grade.

Matthew's primary responsibilities as a TA are to hold office hours for helping with homework and projects, and to grade homework. Given his natural willingness to be helpful, there really is no reason that anyone should not get A's on the homework problems.

Whereas it is a good use of Matthew's talents to help if you become stuck in writing MATLAB code for a homework problem, it definitely would be a waste of his talents to test umpteen MATLAB programs for correctness of output. It is for this reason that I only require each team to submit a single homework solution rather than require every student to submit a solution. The point of homework is to lay foundations for doing projects, not to test you.

Regarding office hours: For each week henceforth, office hours will be posted no later than Sunday evening. You will find the week's office hours under `Course Info`. Oftentimes in place of specific office hours you will find instructions simply to send email to arrange a time to meet.

The following assignment is primarily meant to get everyone up to speed using MATLAB. For students who are rusty in using MATLAB, alongside this assignment is posted a document titled "matlab crash course." (I remark, however, that there are even better documents you can get by googling "matlab crash course," but I am not legally allowed to post those documents.)

Posted with this assignment is a workbook containing weekly (adjusted) closing prices for nine of the SPDR ETFs. These nine ETFs are the oldest of the SPDR's, and cover a broad range of the sectors of the S&P 500.

The first worksheet in the workbook assigns numbers to the weeks, the second lists weekly (adjusted) closing prices of the ETF's, and the third lists volume (but you will not need volume for this assignment).

The sheet on volumes is interesting in that it shows how the ETF's gained traction over time. In the first few years, volume was relatively low, and volatility was relatively high. Also interesting are the prices from the financial crisis of late-summer 2008 until March 2009, big drops in *every* sector (in particular, look around weeks 475-500).

Posted alongside the assignment is a matlab data file (`hw1.mat`) which contains matrices called Price and Volume. The matrix Price contains all entries in the second spreadsheet except not the header row or the first column, that is, it contains only the closing prices of the ETF's, beginning with the week farthest in the past and ending recently. Similarly for the matrix Volume.

This kind of data is readily available at `finance.yahoo.com`. (Once you've chosen a particular ETF or some other asset, click on `historical prices`.)

You have access to data of greater detail using the Bloomberg terminal in Mann Library. However, if you want to use the terminal, you need to register and be given a brief training lesson. If your team wants the finer-grained data, I suggest you register now, not wait. (So many times I've seen projects for which the conception was interesting but the execution went nowhere due to having waited to get data from the terminal until only a couple of weeks before the due date.)

I emphasize: If you are working alone (or even in a team of two), it's not necessarily in your interest to devote the time required to collect (and clean) fine-grained data, because far more important (to me) are your creative thoughts in – and careful execution of – your projects.

---

1) (To be discussed in lecture on Monday, January 30.) Here you'll write a MATLAB program for computing average returns and the covariance matrix. These will play the role of μ and V in Markowitz.

Of course there exist MATLAB commands for computing averages and covariances (for example, `mean` and `cov`), but computing averages and covariances from scratch is a useful MATLAB exercise, and besides, we will want to have weighted averages, in which time periods farther in the past are given less weight.

There will be five inputs:

- `Price`: The matrix of historical prices.
- `trade_date`: The number of the week at the beginning of which you will build a portfolio. At that time, the only closing prices that would be known to you are the ones from earlier weeks.
- `frequency`: The number of weeks between samples. If `frequency` $= 1$, for example, the returns and covariances will be computed weekly, whereas if `frequency` $= 4$, monthly returns and covariances will be obtained.

  In modeling with Markowitz, the data μ and V should be consistent with the length of time for which the portfolio will be held. If the portfolio will be held for one year, for example, it is nonsensical to directly use weekly returns and covariances, because typically, the ratio
  $$\frac{\text{average return}}{\text{standard deviation of returns}}$$
  is vastly smaller for weekly returns than for yearly returns. Directly using weekly returns to decide a portfolio to be held one year greatly overestimates the risk of the portfolio.

  In a few weeks, we'll discuss how to properly "project" returns and covariances from short periods to long periods. This is necessary for assets that have been only recently introduced, and is necessary when the portfolio holding period is long. For now, however, you should think of choosing `frequency` to match the length of the planned holding period.

- `num_samples`: The number of samples. The expected return of asset i will be computed from the weekly closing prices $p_{i,w}$ for weeks

$$w = \texttt{trade\_date} - 1 - \ell \cdot \texttt{frequency} \quad \text{where } \ell = 0, \ldots, \texttt{num\_samples} .$$

  (The number of price samples will thus be $\texttt{num\_samples} + 1$, but the number of return samples will be `num_samples`.)

  The value chosen for `num_samples` always MUST, MUST, MUST exceed the number of assets, preferably by a lot. If it is chosen to be less than the number of assets, the covariance matrix V will not be invertible, and hence in the Markowitz model there will exist portfolios composed of risky assets for which the overall portfolio has no risk. If `num_samples` is only slightly larger than the number of assets, then V is likely to be almost non-invertible, leading the solver to think that some portfolios have far less risk than is actually the case.

  Obviously, when choosing values for `trade_date`, `frequency` and `num_samples`, the relation

$$1 \leq \texttt{trade\_date} - 1 - \texttt{num\_samples} \cdot \texttt{frequency}$$

  must be satisfied.

- `rate_of_decay`: This value will be used to weight sample returns, giving less weight to ones from farther in the past. The input should satisfy

$$0 \leq \texttt{rate\_of\_decay} < 1 .$$

  For two consecutive sample returns, the weight $\texttt{weight}(t)$ given to the earlier sample will differ from the weight given to the later sample according to

$$\texttt{weight}(t) = (1 - \texttt{rate\_of\_decay}) \cdot \texttt{weight}(t+1) .$$

  (What you'll be computing is known as an "exponential moving average.")

  Typically, `rate_of_decay` is chosen close to 0 (say, `rate_of_decay` = .025), especially when using many samples.

  The weights should be treated as a probability measure. In particular, the weights should sum to 1.

The outputs to your program `stats.m` should be the (column) vector `mu` and the covariance matrix `V`.

2) Write a program named `return-risk_profile.m`, which draws the two-dimensional graph on slide 19. Inputs should be `mu` and `V`, the outputs from your program `stats.m`. To test your rendition of `return-risk_profile.m`, use as inputs for `stats.m` the choices `trade_date = 850`, `frequency = 4`, `num_samples = 175`, `rate_of_decay = .001`. Then compare your graph with the one posted under `Content/homework`.

Don't give any time to making your graph be fancy (for example, there's no need to label axes).