Computational Finance with C++

Imperial College London

Department of Computing

Markowitz Model & Rolling Window Back-Testing

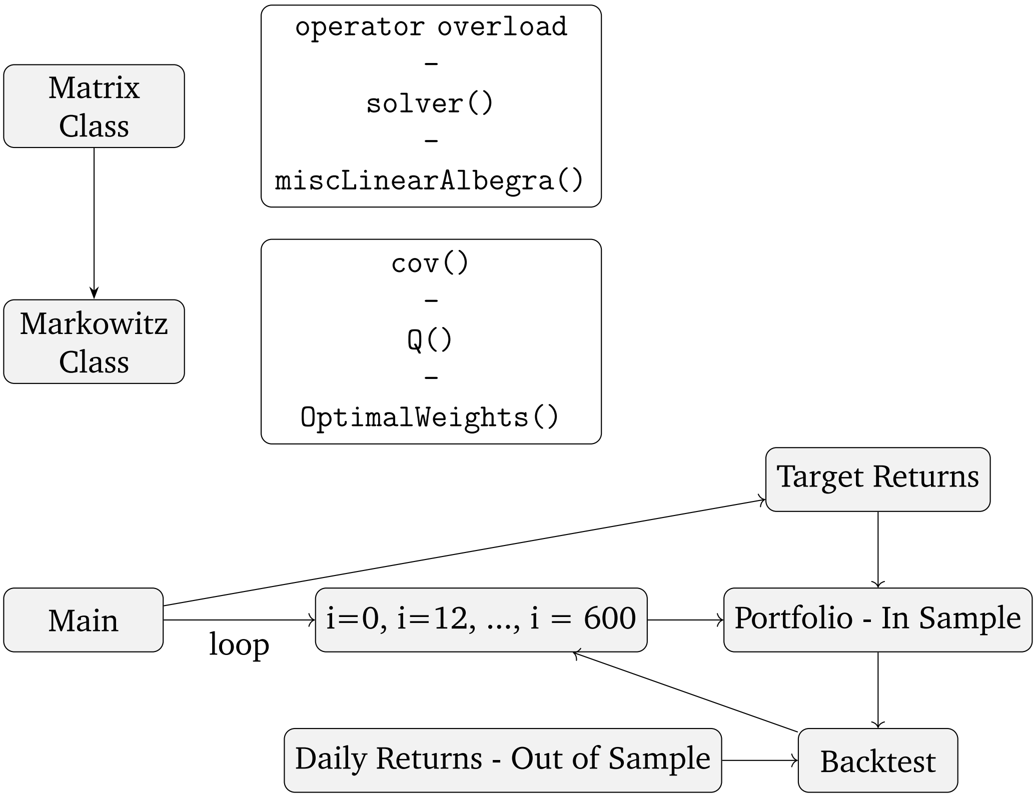
Author:
Edward Peterson (CID: 01502703)

Date: May 23, 2024

1 Software Structure

There was no use of polymorphism.

- read_data.h unchanged
- defined type Vector and Lattice for vector<double> and vector<vector<double>>
- defined class Matrix for holding a lattice and implementing rudimentary linear algebra, multiple constructors e.g. Matrix(rows, columns) or Matrix(Lattice) or Matrix()
  - operator overload for multiplication, addition, subtraction, unary negative and also for scalar equivalent operations
  - operator overload for splcing, along with functionallity for insertion, printing, retrieval, shape etc.
  - Ultimately building towards implementing the Conjugate Gradient Descent Solver.
- implemented numpy like horizontal and vertical stacking of matrices
- Markowitz class for defining a portfolio with optimal asset weights
  - mean() - average returns for each asset over sample period - $\bar{r}_i = \frac{1}{n}\Sigma_{k=1}^{n} r_{i,k}$
  - cov() - covariance of asset returns - $\Sigma_{ij} = \frac{1}{n-1}\sum_{k=1}^{n}(r_{i,k} - \bar{r}_i)(r_{j,k} - \bar{r}_j)$
  - b(double target_return), Q() - vstack(hstack, hstack, hstack)
  - optimal_weights(): $Qx = b$


Matrix Class
Markowitz Class
operator overload
-
solver()
-
miscLinearAlbegra()
cov()
-
Q()
-
OptimalWeights()
Target Returns
Main
loop
i=0, i=12, ..., i = 600
Portfolio - In Sample
Daily Returns - Out of Sample
Backtest