

LOW VOLATILITY MODEL

The Low Vol model is composed of multiple modules, which put together generates the orders to create/rebalance the portfolio on a quarterly basis. The entire model is split into three main components: **1) Basket creation 2) Portfolio Optimization and 3) Order Generation**. The low_vol_model sources data directly from Bloomberg so proper connection to Bloomberg is central for functioning of the model

1) Basket Creation: Basket creation component is composed of 5 modules – a) get_data b) resample c) concat d) z_score and e) basket

a) get_data: The module pulls past 5 years of daily data of a company from Bloomberg and performs data cleaning, manipulation and calculation for all the data fields for that company such as calculating 1yr, 3yr and 5yr std etc. All the columns are renamed to generic names and missing fields/columns are created to make sure the columns of all companies align going forward. Also, other attributes such as id, sector, country, exchange, index weight and currency of that company is pulled from the internal database and put together in a dataframe. – Most of the heavy lifting of company specific data is done in this module.

b) resample: The resample module takes the past 5 years of cleaned data of the company and converts it into quarterly frequency (data for the last day of the quarter). Fills up the “None” values with different “fillna” techniques and only retains the data for the latest quarter that ended. If latest quarter data for a field is not found, it takes the data from the previous quarter. Fields such as ROE, ROIC and Net Debt/EBITDA is not present for the latest quarter at the quarter end-date since the company hasn’t reported the quarter, as a result for these fields the model relies on previous quarter data.

c) concat: This is the module that puts together last quarter data of all companies into one dataframe. The module loops through all the 1,200 companies in our universe (companies in the iShares MSCI World ETF) and for each company calls the get_data and resample modules and appends the company’s data into one dataframe. This completes the pre-processing of the data and is ready for z-score calculation and ranking.

d) z_score: By now we have last quarter data for all companies in our universe. In this module, z-score of every factor for every company is calculated based on sector attributes i.e. sector mean and standard deviation for that factor, to remove sector biases when picking stocks for the basket. Once the factor z-scores for every company are calculated, we weight the z-scores and add/subtract them (based on the factor) to get the final score for a company. Finally, the module also ranks the companies based on the scores. **Z-score is calculated as:**

$$z_{i,f} = \frac{x_{i,f} - \mu_{s,f}}{\sigma_{s,f}}$$

where, i = security, f = raw factor data (10 factors) and s = sector

Score matrix for companies is calculated as:


$$score_{n,1} = Z_{n,w} * F_{w,1}$$

where $Z_{n,w}$ is a (no. of securities by no. of factors) z-score matrix consisting of z-scores of all factors for all companies and $z_{i,f} \in Z_{n,w}$

$F_{w,1}$ is a (no. of factors by 1) fixed factor weight matrix (see on the right)

$score_{n,1}$ is a (no. of securities by 1) score matrix, which contains the scores of all companies

$$F_{w,1} = \begin{bmatrix} -0.15 \\ -0.25 \\ -0.15 \\ -0.15 \\ 0.10 \\ 0.025 \\ 0.025 \\ 0.025 \\ 0.075 \\ -0.05 \end{bmatrix}$$



We are using three categories of factors in the model. Below are the weights, and the matrix above represent these weights:

Volatility (70%)

- 1yr std (x1): 15.0%
- 3yr std (x2): 25.0%
- 5yr std (x3): 15.0%
- 2yr Adj. Beta (x4): 15.0%

Value (15%)

- FCF yield (x5): 10.0%
- B/P (x6): 2.5%
- Dividend yield (x7): 2.5%

Quality (15%)

- ROE (x8): 2.5%
- ROIC (x9): 7.5%
- Net Debt/EBITDA (x10): 5.0%

Simple form of the score equation above for a specific company:

$$\text{Score} = -0.15(x1) - 0.25(x2) - 0.15(x3) - 0.15(x4) + 0.10(x5) + 0.025(x6) + 0.025(x7) + 0.025(x8) + 0.075(x9) - 0.05(x10)$$

e) basket: This module picks the top 6.5% of companies from each sector and completes the basket creation.

2) Portfolio Optimization: Portfolio optimization component is composed of 5 modules:

a) data_collector b) portfolio_constraints c) cash_balance d) std_search (standard deviation search) and e) optimal_portfolio

a) data_collector: data_collector module collects the investable basket from the previous module, gets the tickers and loops through every ticker to get last 3yr daily price data from Bloomberg. All companies are then appended to one dataframe, null values are taken care of and daily returns are calculated. From the daily returns dataframe, a covariance matrix and return matrix is drawn to be used further in the optimization process.

b) portfolio_constraints: this builds in all the security, sector and country level constraints for the optimization model. The model builds in 0.60% - 3.50% max – min range for each security, max 10% sector overweight compared to the benchmark and max 12% country overweight compared to the benchmark. Other essential constraints such as “security weights should equal to 1” are also built in this module. All constraints are represented in matrix form. The matrices are named **G** (matrix representing left-side of all inequality constraints/functions), **h** (right-side of all inequality constraints), **A** (left-side of all equality constraints) and **b** (right-side of all equality constraints)

c) cash_balance: builds in 1.0% of cash at the time of re-balancing as a result every quarter the cash balance of the portfolio goes back to 1.0%.

d) std_search: A recursive module that recursively searches for a portfolio with a target standard deviation from all the optimal portfolios (in the efficient frontier) generated by the optimal_portfolio module. It starts at a target of 6.8% std and if a portfolio with 6.8% std does not exist among optimal portfolios, its incremented 20bps (i.e. target goes up to 7.0% std) and looks for an optimal portfolio generating 7.0% std. The process continues until an optimal portfolio is found.

Note: That sometimes highest std among the optimal portfolios could fall below 6.8% such as during 2016 and 2017 when market volatility was too low. During such periods, the model will break down since search starts from 6.8%. During those time, the model needs to be tweaked (change the required_std variable in the optimal_portfolio module to lower than 6.8%).

e) optimal_portfolio: this module sets everything up and puts all the previous modules together and runs the optimization algo. Leverages python's cvxopt (convex optimization) library for optimization. The optimization model generates an efficient frontier which is the set of all optimal portfolios. All optimal portfolios are then searched using the std_search module (described earlier) to get the optimal investable portfolio (i.e. the portfolio closest to the target std). **Optimization model objective:**

$$\begin{aligned} \text{Minimize} \quad & \rho * w^T Q w - p^T w \\ \text{Subject to:} \quad & G w \leq h \\ & A w = b \end{aligned}$$

where w is the weight matrix for n no. of companies, which is unknown

Q is the Covariance matrix

p is the return matrix

ρ is a multiplier variable that varies from 0.01 to 100,00,000 (at different ρ , obj function changes, which allows the model to derive multiple optimal portfolios with different risk and return level resulting in the efficient frontier)

G and h represents inequality (class) constraint matrices that include security, sector and country constraints

Constraint eqn:

Security constraint: $\varepsilon_i \leq w_i \leq \delta_i$ where is ε_i min weight in a security and is δ_i max weight in a security

Sector constraint: $\Gamma_m : \sum_{i=1}^n w_i \leq w_m$ where Γ_m is a sector m , n is the no. of securities in the sector,

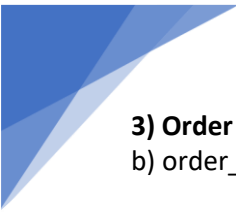
w_i is weight of security i and w_m is max allowable weight for that sector

Country constraint: $\Lambda_k : \sum_{i=1}^l w_i \leq w_k$ where Λ_k is a country k , l is the no. of securities in the country,

w_i is weight of security i and w_k is max allowable weight for that country

A and b represents equality constraint matrix, specifying that sum of all weights should equal to one

Constraint eqn: $\sum_{i=1}^N w_i = 1$ where N is no. of securities in the portfolio



3) Order Generation: The Order Generation component is composed of two modules a) holdings and b) order_generation.

a) holdings: the holdings module does the dollar allocation and calculates the no. of shares to Buy of each security for the upcoming quarter. The model assumes quarter-end price at domestic (local) currency and uses the quarter-end currency rate to convert CAD allocation into domestic (local) currency allocation. The two factors are then used to calculate the no. of shares to Buy. Directly buys from local exchange (no ADRs are used).

b) order_generation: the order_generation module opens up the current_holdings file and matches that with the new portfolio. If a new company comes in “NEW BUY” with the quantity is generated. If an existing company is not present in the new portfolio, “SELL ALL” with the quantity is triggered. For names currently in the portfolio, the difference in quantity between new and existing portfolios is bought or sold.

The model is run at the first day of every quarter and trades are executed during the first day of the quarter.

How the model works and inputs and outputs (main model: low_vol_model)

The modules are built on top of each other, i.e. they rely on each other to work. If one of the modules is broken, the entire model will fall apart. The main model ends with the optimal_portfolio module, which saves three files on the drive (“Archive”) and the Order Generation modules, leverages the saved files to generate final outputs

For eg, once the program is executed, the last module (optimal_portfolio) gets executed, which calls another function and that function calls another function and so on.

INPUTS TO THE MODEL: RUN THE MODEL UPTILL (“optimal_portfolio”)

Four inputs are required for the model:

1) period_5yr: date 5 years back from end of last quarter date in the format (“yyyymmdd”)

For eg, In Jul 1, 2019, period_5yr: “20140630”

2) period_3yr: date 3 years back from end of current quarter date in the format (“yyyymmdd”)

For eg, In Jul 1, 2019, period_3yr: “20160630”

3) cutoff_date: latest quarter end date in the format (“yyyymmdd”)

For eg, in July 1, 2019, cutoff_date: “20190630”

4) rebalancing_term: For quarterly rebalancing use “Q” and for monthly rebalancing use “M”

OUTPUT:

- Three files get saved in the “Archive” folder in the drive, which contains a portfolio file with security names and tickers, optimal weights, sector, country, currency and exchange (file name: port_{cutoff_date}), a quant basket file which contains the tickers we will be investing in and all the raw data, z-scores, scores and rank for all those companies (file name: quant_basket_{cutoff_date}) and the third file contains the quarter-end price of each company in the basket that will be used to calculate the no. of shares to buy (file name: quarterend_prices_{cutoff_date}).

- The order_generation module will output how many shares to buy/sell of each company after comparing with the existing portfolio.