PORTFOLIO OPTIMIZATION CODE USING MEAN-VARIANCE, SINGLE INDEX MODEL

Description of script

The script "portfolio_optimization_mv" aims to provide a simple way of composing a stock portfolio alongside the efficient frontier obtained via the single index model.

Such model assumes that correlation between securities can be explained by a common factor, which is, a benchmark market. With this assumption, then, we can define correlations, as well as expected returns, as functions of the selected market returns and variance.

Adding to that basic concept, I decided to implement an additional step. Because an investor has a single reporting currency, which is the one he keeps accounting with and uses to pay taxes, it is necessary that all assets are normalized to said currency.
Doing so means that variance and returns of foreign stocks for the investor will not only depend on their base performance, but also on the performance of exchange rates between the foreign currency to which the foreign stock is traded (i.e., USD of Apple stocks for an Italian investor) against the investor's one.

I account for that in a separate class called InternationalDiversification. This class adjusts expectations of returns and variances of foreign stock and returns the adjusted dataframe to the main class of EfficientFrontier.

For theory and nomenclatures used in the script, I suggest taking a look at "modern portfolio theory and investment analysis" by Elton, Gruber, Brown & Goetzmann.

COMPONENTS OF SCRIPT

InternationalDiversification class:

Handles the adjustment of variances for each international stock.

EfficientFrontier

Input structure:

- Assets_data → dataframe containing assets data. To build the dataframe you will need ad id for each stock (id doesn't have to be like the one I'm using to make the code work), which will be used throughout the script to identify each asset; then you may also add, but don't need to, symbol and exchange, while the currency at which the stock is traded is mandatory. Also, assets_data has to contain the implied returns column (as per CAPM), filled with NaN values, to later be computed based on up-to-date data of the benchmark market of choice. Lastly, beta (on the benchmark market) and alphas (extra returns above expected CAPM returns) columns are mandatory as well.
- Assets_daily_returns → dataframe with daily returns (suggested at least 2 years of data) of each asset. I used logReturns, which are compatible with arch models used later to compute variances of each asset. The dataframe has as columns the stock ids, and as rows dates of following format: YEAR-month-day.
- Asset_prices → has stock_ids as index, mandatory columns are currency and "close", which is the close price of last reported trading day. Optional columns are symbol and exchange.
- benchmark_market_data → has a market_id as index (the idea is that you get a row from a comprehensive markets table), mandatory columns are symbol, currency, implied returns (for the script, I measured it using foreward looking method, discounting forecast_cfs at a rate that equals their value to current market price).
- benchmark_market_daily_returns → similar to assets_daily_returns, but has data only about the benchmark market.

- Currencies_returns_to_base → still a daily returns dataframe, derived using as pairs each unique foreign currency in assets_data[‘courrency’] column against investor’s currency, so, for example, you will find columns named: USD/EUR, JPY/EUR, GBp/EUR… for a euro investor.
- Rf → is the floating decimal value of the investor’s currency risk free rate (for euro investors, I suggest using German 10/11 years bond rates)