MFE 431 Quantitative Asset Management Problem Set 2

Vinamra Rai

April 28, 2024

1 Risk Parity

To solve the questions in the problem set, I first installed all the necessary libraries and established a connection with the WRDS (Wharton Research Data Services) API. Using the code reference provided by Prof. Herskovic in the lecture, I connected to the WRDS database to retrieve financial data from various CRSP (Center for Research in Security Prices) data tables.

I started by running an SQL query to fetch bond data from the `crspq.tfz_mth` table, which includes bond identifiers (`kycrspid`), calendar dates (`mcaldt`), returns (`tmretnua`), and market values (`tmtotout`). The retrieved data was then processed to convert the date column to a datetime format and rename the columns for better readability.

Next, I ran another SQL query to retrieve T-Bill data from the `crspq.mcti` table, which includes calendar dates (`caldt`) and returns for 30-day (`t30ret`) and 90-day (`t90ret`) T-Bills. Similar to the bond data, the date column was converted to a datetime format and the columns were renamed.

To obtain equity returns data, I executed an SQL query on the `crspq.msf` and `crsp.msenames` tables. This query joined the two tables to retrieve various stock-related data, including identifiers (`permno`, `permco`), dates, share codes (`shrcd`), exchange codes (`exchcd`), returns (`ret`), returns without dividends (`retx`), shares outstanding (`shrout`), prices (`prc`), and adjustment factors (`cfacshr`, `cfacpr`). The data was filtered for records between January 1, 1900, and December 31, 2023, and then sorted by stock identifier (`permno`) and date.

To incorporate delisting returns, I ran another SQL query on the `crspq.msedelist` table, which includes stock identifiers (`permno`), delisting returns (`dlret`), delisting dates (`dlstdt`), and delisting codes (`dlstcd`). The retrieved data was sorted and processed to ensure consistency with the equity returns data.

Throughout the data retrieval process, I performed various data cleaning and preprocessing steps, such as converting data types, adjusting date formats, and resetting indexes. Finally, I merged the delisting returns data with the equity returns data to create a comprehensive dataset for analysis.

1.1 Creating Bond Market Return Portfolios

To solve the first question of constructing the equal-weighted bond market return, value-weighted bond market return, and lagged total bond market capitalization using CRSP Bond data, I defined a function that takes the bond data as input and performs the necessary calculations.

Step 1: Data Cleaning

The function starts by cleaning the input bond data. It removes any rows that contain missing values (NaN) in the *ret* (return) or *me* (market value) columns. This step ensures that the calculations are performed on complete and valid data.

Step 2: Creating Year and Month Columns

To facilitate the grouping of data by year and month, I created two new columns in the bond data DataFrame: *Year* and *Month*. These columns are extracted from the *date* column using the corresponding attributes of the datetime object.

Step 3: Calculating Lagged Market Value

To calculate the value-weighted returns, I created a new column called *lagged_MV* in the bond data. This column is calculated by grouping the data by the bond identifier (*idCRSP*) and then shifting the market value (*me*) column by one row within each group. This effectively assigns the market value from the previous month to each bond.

Step 4: Calculating Equal-Weighted and Value-Weighted Returns

Next, I calculated the equal-weighted and value-weighted returns for each year and month combination. To do this, I grouped the bond data by the *Year* and *Month* columns and performed the following aggregations:

- *Bond_lag_MV*: Sum of the lagged market value (*lagged_MV*) column.
- *Bond_Ew_Ret*: Mean of the return (*ret*) column, representing the equal-weighted return.
- *Bond_Vw_Ret*: Value-weighted return, calculated by multiplying the return (*ret*) column with the corresponding lagged market value (*lagged_MV*), summing the result, and dividing by the sum of lagged market values.

Step 5: Filtering the Results

To ensure that the output includes data from January 1926 onwards, I filtered the results DataFrame to include only the rows where the *Year* column is greater than or equal to 1926.

Step 6: Returning the Results

Finally, the function returns the filtered results DataFrame, which contains the equal-weighted bond market return, value-weighted bond market return, and lagged total bond market capitalization for each year and month combination starting from January 1926.

The resulting DataFrame provides a comprehensive view of the bond market returns and capitalization, enabling further analysis and comparison over the specified time period in the following problems.

1.2 Calculating Excess Returns for Bonds and Stocks

To solve the second question of aggregating stock, bond, and riskless data tables and calculating the lagged market value and excess value-weighted returns for both stocks and bonds, I defined a function that takes the stock data, bond data, and riskless rate data as inputs. For the stock data, I reused the function from Q1 of Problem Set 1.

Step 1: Creating Year and Month Columns

To facilitate the merging of data from different sources, I created 'Year' and 'Month' columns in the stock and riskless rate DataFrames.

Step 2: Merging Stock and Bond Data

To combine the stock and bond data, I performed a left merge on the 'Year' and 'Month' columns. Before merging, I reset the index of the stock DataFrame to convert the datetime index to regular columns. The merged DataFrame contained data from both stocks and bonds, with suffixes '_Stock' and '_Bond' added to the column names to differentiate between them.

Step 3: Merging with Riskless Rate Data

Next, I merged the combined stock and bond DataFrame with the riskless rate DataFrame based on the 'Year' and 'Month' columns. This step added the riskless rate data to the merged DataFrame.

Step 4: Calculating Excess Value-Weighted Returns

To calculate the excess value-weighted returns, I subtracted the riskless rate ('rf30') from the value-weighted returns of stocks ('Stock_Vw_Ret') and bonds ('Bond_Vw_Ret'). The excess returns represented the returns above the riskless rate.

Step 5: Structuring the Final Output

I selected the required columns from the merged DataFrame to structure the final output. The selected columns include 'Year', 'Month', 'Stock_lag_MV' (lagged market value for stocks), 'Stock_Excess_Vw_Ret' (excess value-weighted returns for stocks), 'Bond_lag_MV' (lagged market value for bonds), and 'Bond_Excess_Vw_Ret' (excess value-weighted returns for bonds).

Step 6: Filtering the Results

To ensure that the output includes data from January 1926 onwards, I filtered the results DataFrame to include only the rows where the 'Year' column is greater than or equal to 1926.

Step 7: Returning the Results

Finally, the function returned the filtered results DataFrame, which contained the lagged market value and excess value-weighted returns for both stocks and bonds, along with the corresponding year and month, starting from January 1926.

1.3 Monthly Unlevered and Levered Risk-Parity Portfolio Returns (Asness et al. 2012)

To solve the question of constructing both unlevered and levered risk-parity portfolios that target equal risk contributions from stocks and bonds, I defined a function that takes the portfolio data as input and performs the necessary calculations.

Step 1: Creating Portfolios

I calculated the value-weighted portfolio return above the riskless rate by weighting the excess stock and bond returns by their lagged market values. And then, I constructed the 60-40 portfolio return above the riskless rate by allocating 60% to stock excess returns and 40% to bond excess returns.

Step 2: Computing Inverse Volatilities and Weights

In this step, I computed the inverse volatility for stocks and bonds using a rolling 36-month window of excess returns, shifting by one month to avoid look-ahead bias. The unlevered risk-parity portfolio weights were assigned by setting the weight for each asset class equal to the inverse of its volatility estimate, then normalizing the weights to sum to 1 at each rebalancing.

Step 3: Returns for the Risk Parity Portfolio

I computed the unlevered risk-parity portfolio return above the riskless rate by weighting the stock and bond excess returns by their unlevered risk-parity weights.

Step 4: Matching the Volatilities

Futhermore, in this step, I matched the volatility of the risk-parity portfolio to the realized volatility of the value-weighted excess returns. This involved:

a) Multiplying the inverse volatilities by their respective excess returns.

b) Calculating a scaling factor as the ratio of the value-weighted excess return volatility to the volatility of the inverse-volatility weighted sum.

c) Multiplying the unlevered weights by this scaling factor to get the levered weights.

Step 5: Structuring the Final Output

Calculated the levered risk-parity portfolio return using the stock and bond excess returns weighted by their levered risk-parity weights. I dropped the first 36 months, which had incomplete data for the rolling volatility calculations. As a final step, I structured the final output DataFrame with columns for the date, component returns, volatility estimates, weights, and risk-parity portfolio returns.

1.4 Historical Performance of the Risk Parity, Market, and 60/40 Portfolios

Step 1: Data Filtering and Selection

I started the final exercise by filtering the `Port_Rets` DataFrame to include only the date range from 1929 to 2010, excluding the last 6 months of 2010, as specified in the question.

Step 2: Annualization of Returns and Volatility

I calculated the annualized excess return by multiplying the mean monthly return by 12. Simultaneously, the annualized volatility was calculated by multiplying the standard deviation of monthly returns by the square root of 12.

Step 3: Sharpe Ratio and T-Statistic

The annualized Sharpe Ratio was computed by dividing the annualized excess return by the annualized volatility. Then, I calculated the t-statistic for the annualized excess return.

Step 4: Higher Moments

To compare the higher moments of the distribution, I calculated the skew and kurtosis for the return data.

Step 5: Summary and Formatting

To aggregate the results, I created a summary DataFrame with the calculated statistics, rounded them down to two decimal places and formatted them as percentages where appropriate. Structured the final output DataFrame to match the format of the paper, with specific row and column orders.

The minor differences in the reported values could be due to various factors such as rounding errors or slight variations in the methodology regarding handling of missing data. Despite these negligible differences, the overall results and conclusions remain mostly consistent with the paper.

The key observation is that the Sharpe Ratio of the Risk Parity (Leveraged) portfolio is significantly higher than that of CRSP Stocks, even though its average return is slightly lower. This finding supports the main argument of Asness, Frazzini, and Pedersen (2012) that leverage aversion can lead to superior risk-adjusted returns for portfolios that overweight safer assets, such as the Risk Parity portfolio.

The higher Sharpe Ratio of the Risk Parity portfolio suggests that it offers a more attractive risk-return trade-off compared to a traditional stock-heavy portfolio. By balancing risk contributions across asset classes and applying leverage to the resulting portfolio, investors can potentially achieve better diversification and improved risk-adjusted performance. The minor difference in average returns between the Risk Parity portfolio and CRSP Stocks could be attributed to the data handling methods. Nevertheless, the significantly higher Sharpe Ratio of the Risk Parity portfolio demonstrates its ability to deliver superior risk-adjusted returns, which is the primary focus of the paper's argument.

	Excess Return	**t-stat**	**Volatility**	**Sharpe Ratio**	**Skewness**	**Excess Kurtosis**
CRSP Stocks	6.72	3.17	19.12	0.35	0.22	7.71
CRSP Bonds	1.40	4.32	2.93	0.48	0.22	4.12
Value-Weighted Portfolio	3.51	2.63	12.05	0.29	-0.54	4.54
60/40 Portfolio	4.59	3.55	11.67	0.39	0.23	7.45
RP, Unlevered	2.10	5.08	3.73	0.56	0.08	2.61
RP	6.69	5.02	12.03	0.56	-0.40	1.98