Mathematical Modeling and Algorithm Application
ISSN: 3006-0842 | Vol. 3, No. 1, 2024

Practice of Python in Programming and Optimization of Quantitative Analysis Model of Fixed Income

Yang Li*, Yanchen Zou, Yanda Qian

Xi'an Jiaotong-Liverpool University, Suzhou, China

* **Corresponding author**: Yang Li

Abstract: Python has become an indispensable tool in fixed income trading. This paper introduces the principles and general process of data analysis visualisation by analysing the basic operation and performance of personal investment and finance, combined with data analysis in the era of big data. The library of data analysis tools using Python has become an indispensable tool in fixed income trading. In fixed income trading, it supports fixed income traders to handle tasks such as bond pricing, interest rate modelling, and credit risk analysis Using Python's powerful data processing capabilities, traders can simplify data collection, analysis, and reporting to make better decisions. Based on the Python language, financial data is acquired from different platforms, processed and analysed step-by-step, ensuring that the model remains accurate and stable in different market conditions. Backtesting and ongoing performance evaluations have revealed that Python can be a valuable asset in the dynamic world of fixed income trading by refining trading strategies and effectively managing risk.

Keywords: Python; Quantitative Analysis; Fixed Income.

1. Introduction

Fixed income markets are growing in importance in the investment arena as a key component of asset allocation, risk management and income enhancement strategies, and the diversification of product offerings, the flexibility of trading strategies, and the ever-changing market risks suggest a corresponding increase in market complexity [1]. In this way, effectively analysing and understanding the functioning of fixed income markets has become a central concern for investors and academic researchers alike.

Quantitative analysis of data-centric investments has been widely used in the fixed income field in recent years. Through the construction of mathematical models, market data for in-depth mining and pattern recognition, so as to provide investors with a more accurate and scientific basis for decision-making [2-3].Python in quantitative analysis of fixed income should be in the acquisition and processing of basic data, in-depth to the construction of the model, strategy optimisation, risk management and other aspects. Through systematic combing and practical case demonstration, we explore the specific application and value of Python in fixed income quantitative analysis model programming and optimisation.

2. Application of Python in quantitative analysis of fixed income

2.1. Data acquisition and processing

There are several forms of Python applications when it comes to data collection. Among them, the two most commonly used methods are API interface call and web crawler. API interface call is a stable and efficient way to obtain data. Many financial data providers provide API interfaces, allowing users to automatically obtain the latest market data through Python scripts. For example, key information such as bond price, yield and market index can be easily obtained by calling the corresponding API functions. In addition, web crawler is also a common means to obtain data in Python. For those data sources that don't provide API interface, write a crawler program to grab the relevant information on the web page. Requests library and Beautiful Soup library in Python are common tools for web crawler, which can send HTTP requests and parse HTML pages, thus extracting the required data.

After obtaining the original data, data cleaning, conversion and processing are carried out. Data cleaning is mainly to remove duplicate, wrong or invalid data to ensure the accuracy and integrity of data. Pandas library in Python provides a wealth of data cleaning functions, such as deduplication, filling in missing values, outlier detection and so on. Data conversion is to convert data from one format to another to meet the needs of subsequent analysis.

2.2. Quantization model construction

In the basic framework of fixed income analysis, the yield curve model can describe the relationship between bond yields of different maturities [4-5]. By adjusting the market data to construct a yield curve that accurately reflects the current state of the market, the model is very important for understanding the market dynamics. Under the Python environment, a series of bond yield data with different maturities can be collected by using the computational ability and curve fitting function of libraries such as numpy and scipy; a suitable fitting function is selected according to the characteristics of the data; and algorithms such as scipy. optimize. curve_fit are used to carry out the curve fitting operation and to determine the specific parameters of the yield curve. The yield curve is plotted according to the fitting results, and its shape and trend are analysed in depth for the purpose of realising the potential trend and risk profile of the bond market.

The VaR (Value at Risk) model can also be used to quantify the maximum loss a portfolio may face at a given confidence interval and holding period. In the fixed income market, VaR is considered as one of the key metrics for assessing market risk [6]. The process of realizing VaR model with Python mainly includes the following steps:

Collect historical return data of portfolio.

According to the distribution characteristics of data, choose the appropriate method to calculate VaR.

Use libraries such as numpy and scipy. stats for statistical calculation, and get the VaR value.

Analyze the VaR results and evaluate the market risk level of the portfolio.

In addition to the above two models, there are many other quantitative models widely used in fixed income analysis, such as duration model, convexity model, term structure model of interest rate and so on.

2.3. Strategy optimization and backtesting

Adjust and improve the investment strategy through strategy optimization to improve the income and reduce the risk. The strategy backtesting is an important means to verify the optimization effect, which can evaluate the performance of the strategy in historical data and predict its possible future performance.

Strategy optimization aims to maximize returns and control risks by finding the best portfolio and trading opportunity. This process involves parameter optimization, such as adjusting key parameters such as trading frequency, holding period and stop loss point to find the best combination; Asset allocation optimization, adjusting the proportion of various assets in the portfolio according to the risk and return characteristics of different assets; And risk control, by setting reasonable stop-loss points and take-profit points to control the maximum retracement, to ensure the stability of the strategy in an unfavorable market environment. In the optimisation process, the principle of scientific rigour is always upheld, and the optimisation strategy should not only be based on comprehensive and in-depth data analysis and rigorous statistical validation, but also avoid relying on irrational and subjective judgments that may lead to deviation or distortion of the strategy. The optimisation process needs to be particularly robust to ensure that the optimisation results obtained can maintain consistency and reliability in a changing market environment, and are not limited to the applicability to a particular data set. The principle of simplicity should not be ignored, and model complexity needs to be consciously controlled during the modelling process to avoid overfitting due to over-complexity of the model. In strategy backtesting.

Python tools, with their flexibility and powerful data processing capabilities, show significant advantages and can provide more accurate and efficient support for optimisation. Thanks to its powerful data processing ability and flexible programming characteristics, the backtesting process is not only efficient but also accurate. The basic steps of strategy backtesting with Python include: loading and processing historical data with pandas library, covering key indicators such as price and turnover; According to the optimized strategic logic, the trading algorithm is written, which involves the judgment of buying and selling signals and position management; Simulate the execution strategy on historical data, and record the execution and results of each transaction in detail; After that, the performance of the strategy is comprehensively evaluated by calculating the cumulative income, maximum retracement, Sharp ratio and other indicators. Finally, using matplotlib and other libraries to draw the profit curve and risk control chart, etc., to visually show the performance of the strategy.

3. Case analysis

3.1. Data source

In this case, a representative fixed-income product - government bonds - is selected as the analysis object. The government bond market is large in scale, active in trading, and relatively open and transparent in data, making it convenient for acquisition and analysis.

The data used in this case study covers multiple dimensions of the government bond market, with time spans ranging from the last 3 months to the past 10 years, and covering millions of data records in terms of quantity. The data mainly comes from government bond issuance, transaction, and settlement data regularly released by official institutions such as the Ministry of Finance and the Central Settlement Company. Collect monthly and annual government bond issuance data for the past five years, while obtaining government bond transaction data for each trading day over the past three years. Also, download government bond yield data from the Wind platform over the past 10 years to facilitate long-term trend analysis and strategy back-testing. Various indicators related to government bonds are filtered from the East Money Wealth Choice data to analyze market activity and liquidity.

3.2. Quantitative analysis process based on Python

3.2.1. Data processing

The pandas library of Python is used to clean, transform and integrate the collected national debt data. The code implementation of data loading and preliminary cleaning functions is as follows:

```
Def load_and_clean_data(filepath, date_col='date', parse_dates=True, dropna=True):
    # Load data
    data = pd.read_csv(filepath)

    # Convert a date to a date type
    if parse_dates:
        data[date_col] = pd.to_datetime(data[date_col])

    # Delete the row containing NaN
    if dropna:
        data.dropna(inplace=True)

    return data
```

3.2.2. Model construction

Using Python to build a quantitative model: yield curve model, using polynomial fitting and spline interpolation methods to build the yield curve of national debt, and analyzing the yield relationship of national debt with different maturities. VaR model is based on historical simulation method and parameter method to calculate the maximum possible loss of national debt portfolio in a specific confidence level and holding period.

Among them, the historical simulation method VaR is realized as follows:

```
def historical_var(returns, confidence_level=0.95):
    # Calculate VaR
    return-np.percentile(returns, (1-confidence_level)*100)

# VaR at 95% confidence level
historical_var_95 = historical_var(returns, 0.95)
print(f"95% Confidence Level, Historical VaR:
```

{historical_var_95}")

Parameter method assumes that the rate of return obeys normal distribution, and uses mean and standard deviation to calculate VaR:

```
def parametric_var(returns, confidence_level=0.95):
    mean = np.mean(returns)
    std_dev = np.std(returns)
    # Calculate VaR
    return -norm.ppf(confidence_level) * std_dev - mean

#VaR at 95% confidence level
parametric_var_95 = parametric_var(returns, 0.95)
print(f"95% Confidence Level, Parametric VaR: {parametric_var_95}")
```

3.2.3. Strategy optimization and backtesting

The trading algorithm is written in Python, which generates buying and selling signals according to the preset investment logic and market conditions. The algorithm includes price trend analysis, volume change, and calculation of relative strength index (RSI) and other technical indicators to judge the trading opportunity. In order to effectively manage risk, position ratios are flexibly adjusted based on market volatility and expectations of future returns, aiming to maximise profit potential while preserving principal. A combination of grid search and stochastic search was used to optimise key parameters of the strategy, including core variables such as trading frequency, RSI thresholds and position size. The optimal configuration of each parameter is found by backtesting the historical performance of different parameter combinations. With the help of mean-variance optimisation and the Black-Litterman model, the asset allocation ratios were adjusted to determine the optimal asset mix. On the basis of the optimised parameters, the strategy was simulated using historical data to ensure the feasibility of the strategy in the real market. Throughout the process, potential factors such as transaction costs and slippage were taken into account to make the simulation results closer to the real market environment.

Figure 1 shows the historical performance of the simulated strategy over the past 36 months. As we can see, the overall return curve shows a steady upward trend, and the strategy has achieved a positive cumulative return during this period, which reflects the ability of the strategy to effectively identify and seize profit opportunities under specific investment logic and market conditions. It is worth noting that the return curve is not linear, but rather volatile, suggesting that the strategy's performance has been affected by changes in the market environment. Despite these fluctuations, the overall performance still demonstrates the strategy's robustness and profitability. Over the past 36 months, the simulated strategy has demonstrated solid profitability and good risk control, providing investors with valuable reference information to help them better understand and evaluate the performance of the strategy.

Figure 1 Strategy performance over time

Figure 2 shows that most of the annual returns are concentrated in the range of 5% to 15%, which shows that the strategy can achieve stable positive returns in most cases. The peak (that is, the highest pillar) appears around 10%, which can be regarded as the most possible or expected value of the annual income of the strategy, reflecting the average profitability of the strategy. The strategy shows a stable annual profitability under the simulated data, and the yield is mainly concentrated in the range of 5% to 15%, and an annual return of about 10% is expected. At the same time, the strategy also shows some volatility and there is a risk of negative returns, but the probability of this happening is relatively low.

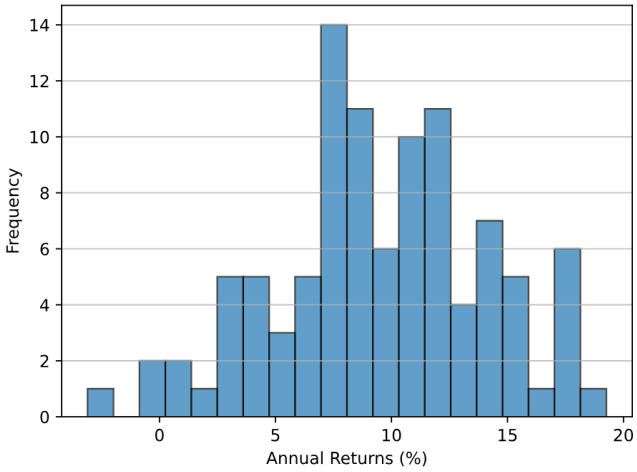

Figure 2 Histogram of annual returns

Python provides a powerful tool for quantitative analysis of fixed income with its powerful data processing ability, rich library function support and flexible programming characteristics. By using Python, we can reveal the market rules more efficiently and make scientific investment strategies. In quantitative analysis, data occupies a central position. This case study involved the collection and processing of extensive government bond market data, providing a solid informational foundation for investment decisions. A data-driven approach to investment decision-making can help investors capture market opportunities more accurately while avoiding potential risks. In the process of continuously improving the strategies through strategy optimisation and backtesting evaluation, it not only enhances the overall performance and robustness of the strategies, but also enables investors to have more confidence in the actual trading operations, thus achieving their investment objectives more effectively.

4. Conclusion

A data-driven approach to investment decision-making enables investors to more accurately identify market opportunities and avoid potential risks. Through strategy optimisation and back-testing evaluation, investment strategies are continuously improved to enhance their

performance and robustness, enabling investors to execute their strategies with greater confidence in actual trading, thus achieving the goal of effectively achieving their investment objectives. In the practice of programming and optimization of fixed income quantitative analysis models, Python language has become the tool of choice in this field due to its excellent data processing capabilities, rich library support and flexible programming features. Through API interface calls and web crawler technology, Python can efficiently collect market data; with the support of pandas and other libraries, it can smoothly clean and transform the data, thus ensuring the accuracy and completeness of the subsequent data analysis. In the model building phase, libraries such as numpy and scipy are used to perform various analytical tasks, including return curve modelling and VaR risk assessment. python shows significant advantages in strategy optimisation and backtesting, and is able to assess the effectiveness of the strategy based on historical data, and present the results of the analysis through visual methods. Studies have shown that Python is very effective at handling large-scale government bond data, constructing quantitative models, and performing backtesting of strategies. Python has become an indispensable tool for quantitative fixed income analysis by dramatically improving the efficiency and accuracy of fixed income market analyses, and by providing investors with solid scientific decision support.

References

[1] Li Jing, Zhang Dong, & Chen Jinhong. (2018). Real estate development investment, economic growth, and transportation infrastructure: An examination based on the panel-VAR model. Journal of Economic Issues Exploration (7), 7.

[2] Zhao Xiaomin, & Tong Jie. (2019). Research on the interactive relationship between China's logistics industry and economic development based on the VAR model. Industrial Technology Economics, 38(3), 8.

[3] Gong Weiwei, Qi Xiangchun, & Pei Shilian. (2018). Research and application of mixed programming methods in Python and R language. Computer Applications and Software, 35(1), 4.

[4] [4] Yang Ke, Guo Yafei, & Tian Fengping. (2023). Cross-market contagion of global systemic financial risks under the impact of economic policy uncertainty shocks: A study based on TVP-FAVAR and TVP-VAR models. Statistical Research, 40(7), 70-84.

[5] He Xiaonian, & Duan Fenghua. (2022). Case analysis of linear regression based on Python. Microcomputer Applications, 38(11), 35-37.

[6] Han Biao, Wang Chao, Gao Xiaoyong, Jiang Yongheng, & Huang Dexian. (2018). Design and application of a solution platform for large-scale complex optimization problems based on Python. Computers and Applied Chemistry, 35(1), 6.