Data Types Retrieved from `yfinance`

Introduction

This document provides an in-depth explanation of the various types of data retrievable from the Python library `yfinance`. Each section describes:

- What the data type represents.
- How it is structured inside the retrieved DataFrame.
- Its relevance to portfolio optimization and financial analysis.

1 Key Data Types

1.1 Adjusted Close (`Adj Close`)

- **Definition:** The adjusted closing price is the stock's closing price after accounting for events like stock splits, dividends, and rights offerings. This provides a more accurate representation of the stock's value over time for analysis.
- **Structure in DataFrame:**
  - Each row corresponds to a trading day.
  - Each column corresponds to the adjusted close price for a specific stock (if multiple tickers are retrieved).
- **Example:**

Date	AAPL	MSFT
2023-01-01	145.5	240.2
2023-01-02	146.3	241.0

- **Relevance:** Used for calculating historical returns, as it reflects the true value of the stock after corporate actions.

1.2 Close (`Close`)

- **Definition:** The closing price is the last price at which the stock traded during a trading session, unadjusted for dividends or splits.
- **Structure in DataFrame:** Same as `Adj Close`, but values are not adjusted for corporate actions.
- **Relevance:** Often used for intraday trading analysis but less useful for long-term historical analysis due to lack of adjustment.

1.3 Open (`Open`), High (`High`), Low (`Low`)

- **Definition:**
  - `Open`: The price at which the stock started trading at the beginning of the session.
  - `High`: The highest price reached during the session.
  - `Low`: The lowest price reached during the session.
- **Structure in DataFrame:** Each column corresponds to one of these metrics for a specific stock.

```
Date          AAPL (Open)    MSFT (Open)
2023-01-01    144.0          239.5
2023-01-02    145.0          240.0
```

- **Relevance:** Useful for understanding price volatility and setting constraints for stop-loss or take-profit strategies.

1.4 Volume (`Volume`)

- **Definition:** The total number of shares traded for the stock during the trading session.
- **Structure in DataFrame:** Each column corresponds to the trading volume for a specific stock.

```
Date          AAPL      MSFT
2023-01-01    1000000   1200000
2023-01-02    1100000   1250000
```

- **Relevance:** Used to assess liquidity and the ability to execute large trades without significant price impact.

1.5 Dividends (`Ticker.dividends`)

- **Definition:** Cash payments made by the company to shareholders as a reward for investing.
- **Structure in DataFrame:** A single-column DataFrame where the index represents the date of the dividend payment.

```
Date          Dividends
2023-02-10    0.22
2023-05-12    0.24
```

- **Relevance:** Contributes to total return calculations, especially for income-focused portfolios.

1.6 Stock Splits (`Ticker.splits`)

- **Definition:** A corporate action that divides existing shares into multiple shares to improve liquidity.
- **Structure in DataFrame:** A single-column DataFrame where the index represents the date of the split and the column contains the split ratio.

```
Date          Splits
2020-08-31    4.0
```

- **Relevance:** Ensures accurate price analysis over time by normalizing historical prices.

1.7 Earnings (`Ticker.earnings`)

- **Definition:** Annual earnings data, including revenue and net income.
- **Structure in DataFrame:**

```
Year          Revenue          Net Income
2022          365817000000     94680000000
```

- **Relevance:** Provides insights into the financial health of a company for screening and selection.

1.8 Real-Time Data (`Ticker.info`)

- **Definition:** A dictionary containing current and fundamental metrics like price, market cap, beta, and sector.
- **Relevance:** Supports dynamic rebalancing and risk management.

2 Structure of Retrieved DataFrames

- Historical data from `yf.download()` or `Ticker.history()`:

Date	Open	High	Low	Close	Adj Close	Volume
2023-01-01	144.0	146.0	143.5	145.5	145.5	1000000
2023-01-02	145.0	147.0	144.5	146.3	146.3	1100000

- Dividends and splits appear as single-column DataFrames with dates as the index.
- Real-time data is returned as a dictionary with key-value pairs.

3 Conclusion

The data types provided by `yfinance` offer comprehensive coverage for financial analysis and portfolio optimization. Understanding the structure and relevance of each data type ensures accurate implementation in portfolio strategies.