finance$_{c}alculator$

Release 0.0.6

Nov 09, 2020

Contents

src.finance_calculator.api.**get_alpha**(*nav_data*, *benchmark_nav_data*, *risk_free_rate=None*, *window=750*, *annualiser=250*, *tail=True*)

Alpha describes a strategy's ability to beat the market, or it's "edge." Alpha is thus also often referred to as "excess return" or "abnormal rate of return," which refers to the idea that markets are efficient, and so there is no way to systematically earn returns that exceed the broad market as a whole. Alpha is often used in conjunction with beta (the Greek letter β), which measures the broad market's overall volatility or risk, known as systematic market risk. (Investopedia)

Parameters

- **nav_data** –
- **benchmark_nav_data** –
- **risk_free_rate** – float
- **window** – int
- **annualiser** – int
- **tail** – bool

Returns

src.finance_calculator.api.**get_beta**(*nav_data*, *benchmark_nav_data*, *risk_free_rate=None*, *window=750*, *annualiser=250*, *tail=True*)

Beta is a measure of the volatility—or systematic risk—of a security or portfolio compared to the market as a whole. Beta is used in the capital asset pricing model (CAPM), which describes the relationship between systematic risk and expected return for assets (usually stocks). CAPM is widely used as a method for pricing risky securities and for generating estimates of the expected returns of assets, considering both the risk of those assets and the cost of capital. (Investopedia)

Parameters

- **nav_data** –
- **benchmark_nav_data** –
- **risk_free_rate** – float
- **window** – int
- **annualiser** – int
- **tail** – bool

src.finance_calculator.api.**get_downside_capture**(*nav_data*, *benchmark_nav_data*, *risk_free_rate=None*, *window=750*, *annualiser=250*, *tail=True*)

The down-market capture ratio is a statistical measure of an investment manager's overall performance in down-markets. It is used to evaluate how well an investment manager performed relative to an index during periods when that index has dropped. The ratio is calculated by dividing the manager's returns by the returns of the index during the down-market and multiplying that factor by 100. (Investopedia)

Parameters

- **nav_data** –
- **benchmark_nav_data** –
- **risk_free_rate** – float
- **window** – int
- **annualiser** – int
- **tail** – bool

Returns

src.finance_calculator.api.**get_drawdown**(*nav_data*, *window=750*, *annualiser=250*, *tail=True*)

A drawdown is a peak-to-trough decline during a specific period for an investment, trading account, or fund. A drawdown is usually quoted as the percentage between the peak and the subsequent trough. If a trading account has $10,000 in it, and the funds drop to $9,000 before moving back above $10,000, then the trading account witnessed a 10% drawdown. (Investopedia)

window is about the rolling window for which the calculation is to be done annualiser is an int which can be understood as the number of entries we have in a year. We are using it to annualise values ans

Parameters

- **nav_data** –
- **window** – int
- **annualiser** – int
- **tail** – bool

Returns

src.finance_calculator.api.**get_ratio_calculator**(*nav_data*, *benchmark_nav_data=None*, *risk_free_rate=None*, *annualiser=250*)

returns a ratio calculator instance which can be used to call functions as below:

```
>>> import finance_calculator as fc
>>> rc = fc.get_ratio_calculator(nav_data, benchmark_nav_data)
>>> beta_df = rc.get_beta(window=250*3)
>>> alpha_df = rc.get_alpha(window=250*3)
```

benefit is that the data pre processing would not happen multiple time if you have to get multiple ratios on the same data-set. You need to pass window value separately for each function

Parameters

- **nav_data** –
- **benchmark_nav_data** –
- **risk_free_rate** – float
- **annualiser** – int

Returns

src.finance_calculator.api.**get_sharpe**(*nav_data*, *benchmark_nav_data*, *risk_free_rate=None*, *window=750*, *annualiser=250*, *tail=True*)

The Sharpe ratio was developed by Nobel laureate William F. Sharpe and is used to help investors understand the return of an investment compared to its risk. The ratio is the average return earned in excess of the risk-free rate per unit of volatility or total risk. Volatility is a measure of the price fluctuations of an asset or portfolio. (Investopedia)

Parameters

- **nav_data** –
- **benchmark_nav_data** –
- **risk_free_rate** – float

- **window** – int
- **annualiser** – int
- **tail** – bool

Returns

src.finance_calculator.api.**get_sortino**(*nav_data*, *benchmark_nav_data*, *risk_free_rate=None*, *window=750*, *annualiser=250*, *tail=True*)

The Sortino ratio is a variation of the Sharpe ratio that differentiates harmful volatility from total overall volatility by using the asset's standard deviation of negative portfolio returns—downside deviation—instead of the total standard deviation of portfolio returns. The Sortino ratio takes an asset or portfolio's return and subtracts the risk-free rate, and then divides that amount by the asset's downside deviation. (Investopedia)

Parameters

- **nav_data** –
- **benchmark_nav_data** –
- **risk_free_rate** – float
- **window** – int
- **annualiser** – int
- **tail** – bool

Returns

src.finance_calculator.api.**get_treynor**(*nav_data*, *benchmark_nav_data*, *risk_free_rate=None*, *window=750*, *annualiser=250*, *tail=True*)

The Treynor ratio, also known as the reward-to-volatility ratio, is a performance metric for determining how much excess return was generated for each unit of risk taken on by a portfolio. (Investopedia)

Parameters

- **nav_data** –
- **benchmark_nav_data** –
- **risk_free_rate** – float
- **window** – int
- **annualiser** – int
- **tail** – bool

Returns

src.finance_calculator.api.**get_upside_capture**(*nav_data*, *benchmark_nav_data*, *risk_free_rate=None*, *window=750*, *annualiser=250*, *tail=True*)

The up-market capture ratio is the statistical measure of an investment manager's overall performance in up-markets. It is used to evaluate how well an investment manager performed relative to an index during periods when that index has risen. The ratio is calculated by dividing the manager's returns by the returns of the index during the up-market and multiplying that factor by 100. (Investopedia)

Parameters

- **nav_data** –
- **benchmark_nav_data** –

- **risk_free_rate** – float
- **window** – int
- **annualiser** – int
- **tail** – bool

Returns

src.finance_calculator.api.**get_volatility**(*nav_data*, *window=750*, *annualiser=250*, *tail=True*)

Volatility is a statistical measure of the dispersion of returns for a given security or market index. In most cases, the higher the volatility, the riskier the security. Volatility is often measured as either the standard deviation or variance between returns from that same security or market index. (Investopedia)

Parameters

- **nav_data** –
- **window** – str
- **annualiser** – int
- **tail** – bool

Returns

src.finance_calculator.api.**get_xirr**(*cashflows*) → int

Returns Excel style xirr IRR: The internal rate of return is a metric used in financial analysis to estimate the profitability of potential investments. The internal rate of return is a discount rate that makes the net present value (NPV) of all cash flows equal to zero in a discounted cash flow analysis. IRR calculations rely on the same formula as NPV does. XIRR is used when the cash flow model does not exactly have annual periodic cash flows. (Investopedia)

Parameters **cashflows** –

Returns int

CHAPTER 1

Overview

docs	
tests	
package	

A simple python tool for calculating ratios used to measure portfolio performance. Ratios include alpha, beta, sharpe, volatility, upside capture, downside capture, sortino ratio, treynor ratio, drawdown etc.

It also can be used to calculating portfolio returns like XIRR. (twirr, holding period return etc. will be added).

The tool is largely based on pandas and numpy and is capable of giving continuous (rolling) values of ratios wherever required in the form of a pandas dataframe. All data (portfolio/ navs/ market) needs to be passed in arguments based on the function getting called.

For example - XIRR can be calculated from portfolio cashflows [(date, amount)]. - Sharpe ratio will need scheme/portfolio nav [(date, nav)]. - Alpha will need both scheme nav as well as benchmark nav.

For definitions of above terms, check Investopedia. You can find the examples of few of these ratios here. https://www.valueresearchonline.com/funds/197/sbi-large-and-midcap-fund

- Free software: BSD 2-Clause License

1.1 Installation

```
pip install finance-calculator
```

You can also install the in-development version with:

```
pip install https://github.com/sprksh/finance-calculator/archive/master.zip
```

1.2 Documentation

https://finance-calculator.readthedocs.io/

1.3 Development

To run all the tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

Windows	`set PYTEST_ADDOPTS=--cov-append` `tox`
Other	`PYTEST_ADDOPTS=--cov-append tox`

Values Reference:

https://www.valueresearchonline.com/funds/197/sbi-large-and-midcap-fund

CHAPTER 2

Installation

At the command line:

```
pip install finance-calculator
```

CHAPTER 3

Usage

To use finance_calculator in a project:

```
>>> import finance_calculator as fc

>>> drawdown = fc.get_drawdown(scheme_data)
>>> volatility = fc.get_volatility(scheme_data)
>>> sharpe = fc.get_sharpe(scheme_data, benchmark_data)
>>> sortino = fc.get_sortino(scheme_data, benchmark_data)
>>> treynor = fc.get_treynor(scheme_data, benchmark_data)
>>> alpha = fc.get_alpha(scheme_data, benchmark_data)
>>> beta = fc.get_beta(scheme_data, benchmark_data)
>>> upside_capture = fc.get_upside_capture(scheme_data, benchmark_data)
>>> downside_capture = fc.get_downside_capture(scheme_data, benchmark_data)
```

If you want only current value of a given ratio, you can use `tail=True` as a keyword argument in all of these functions. With `tail=False` it will give a pandas dataframe with values in a rolling window fashion.

By default it is assumed that data is for 250 days of a year. Also rolling window is taken to be 3 years by default. It can be used as `fc.get_sharpe(scheme_data, benchmark_data, window=500, annualiser=250)` for 2 years rolling data. Annualiser is meant for annualising the values. If your data contains 365 days in a year you should pass `annualiser=365`. Normally, if it contains only working days, it will by default take `annualiser=250`.

Instead of benchmark_nav_data, an annual value of risk_free_rate can be given. The calculator can use it in place of benchmark_nav_data. Internally, it will create a benchmark nav df from the risk_free_rate. The value can passed as `fc.get_sharpe(scheme_data, risk_free_rate=0.05)` for 5% of annual risk free returns.

Also, if multiple out of these are needed for one set of data only, you can get a ratio calculator instance and call the functions on that:

```
>>> import finance_calculator as fc
>>> rc = fc.get_ratio_calculator(nav_data, benchmark_nav_data)
>>> beta_df = rc.get_beta(window=250*3)
>>> alpha_df = rc.get_alpha(window=250*3)
```

(continues on next page)

(continued from previous page)

```
# similarly other ratios can be called
# window needs to be passed for rolling window period in calculations.
```

This ensures that pre-processing is reduced for the data thus improving the performance.

The scheme data and the benchmark data can either be a pandas dataframe or list of tuples: (date, nav).

Also you can use it to calculate xirr:

```
>>> import finance_calculator as fc
>>> cashflow_data = [
    (datetime.date(2020, 3, 1), 10000),
    (datetime.date(2020, 4, 1), 10000),
    (datetime.date(2020, 5, 1), 10000),
    (datetime.date(2020, 6, 1), 10000),
    (datetime.date(2020, 7, 1), 10000),
    (datetime.date(2020, 8, 1), -60000),
]
>>> xirr = fc.get_xirr(cashflow_data)
```

CHAPTER 4

Reference

4.1 finance_calculator

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

5.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.2 Documentation improvements

finance_calculator could always use more documentation, whether as part of the official finance_calculator docs, in docstrings, or even on the web in blog posts, articles, and such.

5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/sprksh/finance-calculator/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

5.4 Development

To set up *finance_calculator* for local development:

1. Fork finance_calculator (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:YOURGITHUBNAME/finance_calculator.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes run all the checks and docs builder with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

CHAPTER 6

Authors

- Surya Prakash - https://sprksh.github.io/

CHAPTER 7

Changelog

7.1 0.0.0 (2020-07-29)

- First release on PyPI.

7.2 0.0.5 (2020-11-09)

- Verified values in ratios calculator

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

s

Index

F

G

S