The Journal of Open Source Software

PyPortfolioOpt: portfolio optimization in Python

Robert Andrew Martin[1]

1 University of Cambridge

DOI: 10.21105/joss.03066

Software

- Review
- Repository
- Archive

Editor: Vissarion Fisikopoulos

Reviewers:

- @omendezmorales
- @SteveDiamond

Submitted: 25 February 2021
Published: 07 May 2021

Summary

Portfolio construction is a critically important aspect of investment management. Even after an investor selects a set of assets or return streams to invest in, it is a nontrivial task to decide how much should be allocated to each. The expected return of the asset is certainly an important factor, but the investor may also wish to consider the investment risks and the co-dependence of asset returns.

Modern Portfolio Theory, introduced by Markowitz (1952), presents a mathematical framework for maximizing a portfolio's expected returns subject to a risk constraint (measuring risk with the covariance matrix of asset returns). While optimization problems are difficult in general, many portfolio optimization tasks can be framed as **convex optimization** problems, inviting the use of a large body of theory and several efficient solving routines (Boyd & Vandenberghe, 2004).

PyPortfolioOpt is a python package that implements financial portfolio optimization techniques, including classical mean-variance optimization (MVO) methods, Black-Litterman allocation (Black & Litterman, 1991), and modern methods such as the machine learning-inspired Hierarchical Risk Parity algorithm (López de Prado, 2016).

PyPortfolioOpt is currently being used by several financial services companies; it has been downloaded over 160,000 times, cited in academic publications (Jansen, 2020; Snow, 2020), and used in numerous online courses and tutorials (Putkov, 2019; Werger, 2021).

Statement of need

There are several open-source solvers for convex optimization problems (most of which have python interfaces), for example, CVXOPT (Anderson et al., 2021), OSQP (Stellato et al., 2020) and ECOS (Domahidi et al., 2013). However, these solvers require the user to write their problem in a particular canonical form, creating a barrier to entry for those lacking the prerequisite technical background.

Domain-specific languages like CVXPY (Diamond & Boyd, 2016) offer a significant improvement in usability. CVXPY allows users to specify their convex optimization problem in intuitive syntax; it then identifies the most suitable solver and rewrites the problem in the appropriate canonical form (Agrawal et al., 2018). In the example below, CVXPY is used to maximize the return of a long-only portfolio subject to the constraint that portfolio volatility may not exceed 20%.

```
import cvxpy as cp

mu = ... # N x 1 vector of expected returns
S = ... # N x N covariance matrix
```

```
w = cp.Variable(N)  # weights to optimize
problem = cp.Problem(
    objective=cp.Maximize(w @ mu),
    constraints=[
        w >= 0,
        cp.sum(w) == 1,
        cp.quad_form(w, S) <= 0.20 ** 2,
    ],
)
problem.solve()
weights = w.value
```

CVXPY requires the user to know the mathematical formulation of their optimization problem and to construct the appropriate expressions from CVXPY atomic functions (e.g. `cvxpy.sum` and `cvxpy.quad_form` above).

PyPortfolioOpt was built on the belief that there are many investors who understand the broad concepts related to portfolio optimization (i.e. they know their objectives and constraints) but are either unable or unwilling to solve the mathematical optimization problem. To that end, PyPortfolioOpt seeks to abstract away as much of the mathematics as possible. Contrast the previous CVXPY-based script to the PyPortfolioOpt solution to the same problem:

```
from pypfopt import EfficientFrontier

mu = ... # N x 1 vector of expected returns
S = ... # N x N covariance matrix
ef = EfficientFrontier(mu, S, weight_bounds=(0, 1))
weights = ef.efficient_risk(target_volatility=0.20)
```

Prior to the release of PyPortfolioOpt, there were several implementations of portfolio optimization routines in Python. However, to the best of our knowledge, PyPortfolioOpt was the first project offering an API for general portfolio optimization (i.e. a library rather than a script).

Methods

A full discussion of the package's functionality may be found on the documentation website (Martin, 2021) – the documentation is open-source and hosted by ReadTheDocs. This notwithstanding, we provide a brief survey of the methods available in PyPortfolioOpt v1.4.1:

- Expected returns: simple methods for estimating expected asset returns
- Risk models: methods for estimating the covariance matrix of asset returns, such as Ledoit-Wolf shrinkage (Ledoit & Wolf, 2004)
- Mean-variance optimization with a flexible API for adding constraints
- General efficient frontier:
  - Mean-semivariance optimization (Estrada, 2008; Markowitz et al., 2020)
  - Mean-CVaR optimization (Rockafellar & Uryasev, 2000)
  - Support for custom optimization problems, e.g minimizing tracking error
- Black-Litterman allocation (Black & Litterman, 1991)
- Hierarchical Risk Parity (López de Prado, 2016)
- Critical Line Algorithm (Bailey & López de Prado, 2013)

- Post-processing: helper methods for converting optimal weights into a discrete allocation
- Plotting routines to visualise the efficient frontier

In the investment management industry, market participants are often incentivised to keep their work proprietary to reduce alpha decay. To that end, PyPortfolioOpt has been designed with modularity in mind, such that users can "plug-in" their own innovations when appropriate while deferring to standard methods elsewhere. PyPortfolioOpt integrates seamlessly with pandas dataframes (McKinney, 2010) and NumPy arrays (Harris et al., 2020), which are commonly used in data analysis. Figure 1 summarises the overall workflow of PyPortfolioOpt:

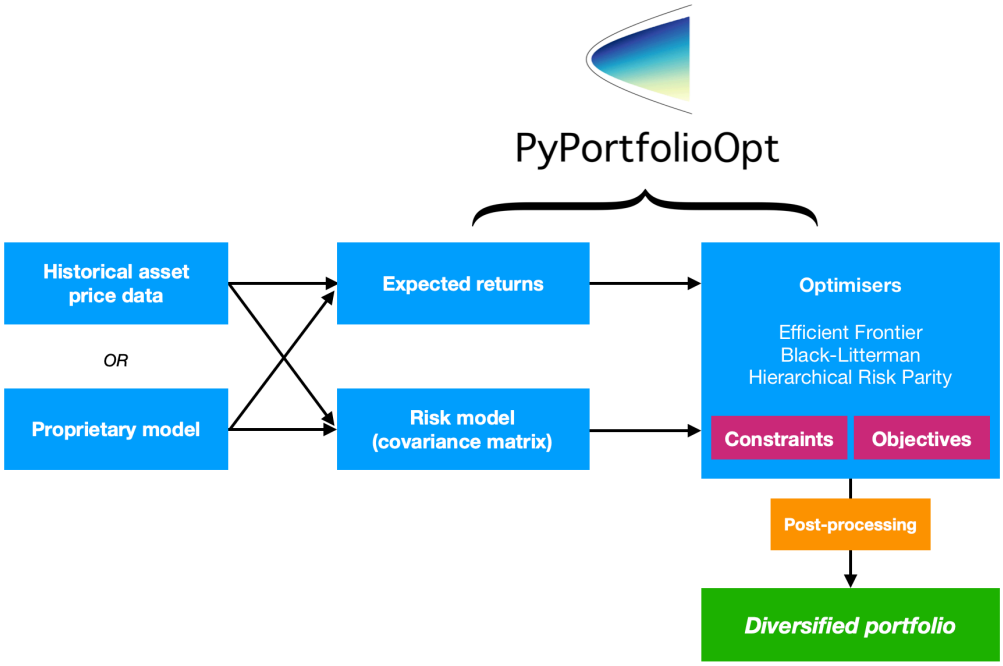

Figure 1: PyPortfolioOpt's modular design allows the package to be integrated with proprietary risk models, return estimates, and constraints.

PyPortfolioOpt is actively maintained and developed. The current feature roadmap includes conditional drawdown optimization (Chekhlov et al., 2005), risk parity portfolios (Spinu, 2013), higher moment optimization (Harvey et al., 2010), and factor models.

Acknowledgements

We would like to thank the following individuals for their contributions to the package's functionality and usability (in no particular order): Philipp Schiele, Nicolas Knudde, Felipe Schneider, Carl Peasnell, Rich Caputo, Dingyuan Wang, Pat Newell, Aditya Bhutra, Thomas Schmelzer, and any future contributors. The CVXPY team, in particular Steven Diamond, have been ever-supportive.

We further acknowledge Marcos López de Prado, whose code for the Critical Line Algorithm and Hierarchical Risk Parity has been used with permission.

Finally, we are grateful to all of the PyPortfolioOpt's users. Their comments and feedback have been instrumental in the improvement of the package.

References

Agrawal, A., Verschueren, R., Diamond, S., & Boyd, S. (2018). A Rewriting System for Convex Optimization Problems. *Journal of Control and Decision*, *5*(1), 42–60.

Anderson, M. S., Dahl, J., & Vandenberghe, L. (2021). *CVXOPT: A Python package for convex optimization*. http://cvxopt.org/

Bailey, D. H., & López de Prado, M. (2013). An Open-Source Implementation of the Critical-Line Algorithm for Portfolio Optimization. *Algorithms*, *6*(1), 169–196. https://doi.org/10.3390/a6010169

Black, F., & Litterman, R. B. (1991). Asset Allocation. *The Journal of Fixed Income*, *1*(2), 7–18. https://doi.org/10.3905/jfi.1991.408013

Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press. ISBN: 0521833787

Chekhlov, A., Uryasev, S., & Zabarankin, M. (2005). Drawdown Measure in Portfolio Optimization. *International Journal of Theoretical and Applied Finance*, *8*(01), 13–58. https://doi.org/10.1142/S0219024905002767

Diamond, S., & Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, *17*(83), 1–5.

Domahidi, A., Chu, E., & Boyd, S. (2013). ECOS: An SOCP solver for embedded systems. *European Control Conference (ECC)*, 3071–3076. https://doi.org/10.23919/ECC.2013.6669541

Estrada, J. (2008). Mean-Semivariance Optimization: A Heuristic Approach. *Journal of Applied Finance*, *18*(1). https://doi.org/10.2139/ssrn.1028206

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., R'ıo, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Harvey, C., Liechty, J., Liechty, M., & Muller, P. (2010). Portfolio selection with higher moments. *Quantitative Finance*, *10*(5), 469–485. https://EconPapers.repec.org/RePEc:taf:quantf:v:10:y:2010:i:5:p:469-485

Jansen, S. (2020). *Machine Learning for Algorithmic Trading* (2nd ed.). Packt Publishing. ISBN: 9781839217715

Ledoit, O., & Wolf, M. (2004). Honey, I shrunk the sample covariance matrix. *The Journal of Portfolio Management*, *30*(4), 110–119. https://doi.org/10.3905/jpm.2004.110

López de Prado, M. (2016). Building Diversified Portfolios that Outperform Out of Sample. *The Journal of Portfolio Management*, *42*(4), 59–69. https://doi.org/10.3905/jpm.2016.42.4.059

Markowitz, H. (1952). Portfolio Selection. *Journal of Finance*, *7*(1), 77–91. https://EconPapers.repec.org/RePEc:bla:jfinan:v:7:y:1952:i:1:p:77-91

Markowitz, H., Starer, D., Fram, H., & Gerber, S. (2020). Avoiding the Downside: A Practical Review of the Critical Line Algorithm for Mean–Semivariance Portfolio Optimization. In *HANDBOOK OF APPLIED INVESTMENT RESEARCH* (pp. 369–415). World Scientific Publishing Co. Pte. Ltd. https://doi.org/10.1142/9789811222634_0017

Martin, R. A. (2021). *PyPortfolioOpt documentation*. https://pyportfolioopt.readthedocs.io/en/latest/

McKinney, Wes. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

Putkov, A. (2019). Portfolio optimization in Modern Portfolio Theory. In *Refinitiv Developer Community*. Refinitiv. https://developers.refinitiv.com/en/article-catalog/article/portfolio-optimization-modern-portfolio-theory

Rockafellar, R., & Uryasev, S. (2000). Optimization of Conditional Value-At-Risk. *Journal of Risk*, *2*, 21–42.

Snow, D. (2020). Machine Learning in Asset Management - Part 2: Portfolio Construction - Weight Optimization. *The Journal of Financial Data Science*, *2*(2), 17–24. https://doi.org/10.3905/jfds.2020.1.029

Spinu, F. (2013). An Algorithm for Computing Risk Parity Weights. *Available at SSRN 2297383*. https://doi.org/10.2139/ssrn.2297383

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., & Boyd, S. (2020). OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, *12*(4), 637–672. https://doi.org/10.1007/s12532-020-00179-2

Werger, C. (2021). Introduction to Portfolio Analysis in Python. In *DataCamp*. https://www.datacamp.com/courses/introduction-to-portfolio-analysis-in-python