

qis: reproducible performance analytics and portfolio backtesting in Python

Artur Sepp ¹

¹ LGT Bank (Schweiz) AG, Zürich, Switzerland

DOI: [10.21105/joss.11400](https://doi.org/10.21105/joss.11400)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Daniel S. Katz](#) 

Reviewers:

- [@alex09x](#)
- [@robertmartin8](#)
- [@tschm](#)

Submitted: 29 July 2026

Published: 18 October 2026

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Quantitative portfolio managers and investors evaluate a strategy by simulating a portfolio that implements it and reporting its risk and return. Using the target weights of a strategy and the prices of its assets, `qis` simulates the portfolio's net asset value. It then measures the portfolio against a benchmark and renders the result as a factsheet. In addition to the backtesting engine, the package provides building blocks for systematic strategies and reports, including exponentially weighted covariance, bootstrap resampling, risk measures, stress testing, regime-conditional statistics, currency hedging, and return unsmoothing.

Signal generation and portfolio construction remain with the researcher. The `qis` library reconstructs holdings from the target weights and applies explicit accounting and reporting conventions.

The `qis` package is the base layer of a public stack of quantitative finance packages, whose other members are `optimalportfolios`, `trendfollowing`, and `privateassets`. The package depends on NumPy ([Harris et al., 2020](#)), SciPy ([Virtanen et al., 2020](#)), pandas ([McKinney, 2010](#)), statsmodels ([Seabold & Perktold, 2010](#)), Matplotlib ([Hunter, 2007](#)), seaborn ([Waskom, 2021](#)), Numba ([Lam et al., 2015](#)), openpyxl, and PyYAML, runs on Python 3.10 to 3.14 and is released under the MIT licence.

Statement of need

The reported performance of a portfolio depends on conventions that are rarely stated. Portfolio volatility depends on the return frequency, and a Sharpe ratio depends on whether a risk-free rate is subtracted and on which rate. A backtested return depends on whether holdings drift between rebalancings or are reset every period. Two implementations of the same target weights can therefore report different performance statistics without either containing an error.

The target users of `qis` are researchers and portfolio managers who generate weights with their own models. The `qis` package settles each convention once and applies it to every factsheet it generates. The caller states the return convention as an argument, and the reporting frequency appears on every rendered output. Three Sharpe conventions, with compounded, log, and arithmetic numerators, are named and selected explicitly.

State of the field

Table 1: Documented interfaces of related Python packages, consulted on 11–14 October 2026.

Package	Documented input	Documented output
pyfolio-reloaded (pyfolio-reloaded contributors, 2026)	returns, optionally positions and transactions	tear sheets and round-trip analysis
bt (bt contributors, 2026)	target weights through WeighTarget in an Algo stack	statistics, weights, positions, and turnover
cvxportfolio (Boyd et al., 2017)	a trading policy, such as FixedWeights for supplied target weights, and market data	value, weights, holdings, turnover, and statistics
cvxsimulator (Schmelzer, 2026)	prices, with units, weights, or cash positions set in a loop over time	a portfolio object tracking units, cash and profit, with a snapshot plot
vectorbt (vectorbt contributors, 2026)	orders, including target portfolio percentages, signals, or an order-generating callback	order and trade records, statistics, and plots
zipline-reloaded (zipline-reloaded contributors, 2026)	orders placed from a callback on each bar, including target portfolio percentages	a daily performance frame
backtrader (backtrader contributors, 2026)	orders placed from a Strategy callback on each bar, including target portfolio percentages	strategy instances with analysers, observers, and plots
Riskfolio-Lib (Riskfolio-Lib contributors, 2026)	returns and optimisation constraints	weights, reports, and plots
skfolio (Nicolini et al., 2026)	returns fitted through a scikit-learn interface	weights, cross-validation, risk measures, and plots
quantstats (QuantStats contributors, 2026)	returns	metrics, plots, and HTML tear sheets
empyrical-reloaded (empyrical-reloaded contributors, 2026)	returns	return and risk metrics
ffn (ffn contributors, 2026)	prices	analytics, with simulation referred to bt
PyPortfolioOpt (Martin, 2021)	prices, or expected returns and a risk model	weights, discrete allocation, performance, and plots

The `qis` backtester accepts dated target weights and asset prices directly, with optional per-instrument costs and carry. For strategies expressed in this form, researchers can evaluate allocations generated by their own models without implementing a strategy callback or policy object. Target-weight support is shared with other packages in [Table 1](#): `vectorbt` supports percentage targets, and `zipline-reloaded` and `backtrader` provide target-percent order methods.

The contribution of `qis` is the integration of held-unit accounting, instrument attribution, and comparative reporting under explicit conventions. The simulation returns a `PortfolioData` object carrying net asset value, held units, realised weights, realised costs, and instrument profit and loss. Factsheets and comparative reports are rendered from it. The chart, the summary table, and the factsheet of one strategy therefore show one Sharpe ratio under one stated convention. The analytics packages `pyfolio-reloaded`, `quantstats`, `empyrical-reloaded`,

and `ffn` measure a portfolio from its history of returns. That history does not record the holdings or the trading costs incurred, so these packages cannot show which holding earned or lost the money unless holdings are supplied separately. The closest package is `cvxportfolio`, whose `FixedWeights` policy trades to supplied target weights and holds the positions between them. Its documented backtest result reports value, weights, holdings, turnover, and total profit, within a framework built around optimisation policies and cost models.

We built a separate library, instead of extending `bt`, `ffn`, or `cvxportfolio`, because our research packages exchange one result object. The `bt` package already accepts target weights, and `ffn` already supplies analytics. The `qis` package combines holdings, costs, attribution, and comparative reports, and its callers adopt neither an Algo stack nor a policy framework. The user keeps the freedom to generate allocation weights in their own strategy code. The trade-off is that `qis` ships no optimiser and no signal library of its own.

The integrated analytics include return unsmoothing for appraisal-based valuations (Geltner, 1993; Getmansky et al., 2004), regime-conditional reporting by benchmark return quantile (Sepp & Dézeraud, 2019), and paired resampling of aligned factor and residual panels with the stationary bootstrap of Politis & Romano (1994). We do not claim these methods as unique. R's `PerformanceAnalytics` implements the first-order unsmoothing of Geltner (1993; Peterson & Carl, 2026), and the `StationaryBootstrap` class of `arch` resamples several aligned arrays with one index draw (Sheppard & others, 2026). The `qis` unsmoothing extends the first-order case to any lag order, estimated once for the sample or over an exponentially weighted rolling window. Its bootstrap resamples returns and recomponds them into price panels.

Software design

The backtester is 460 lines, so its accounting can be audited line by line. The backtester converts target weights to units at each rebalancing, holds those units until the next one, applies costs and carry, and returns the result. Holding units lets the realised weights drift with prices, whereas multiplying each period's returns by fixed target weights would assume rebalancing in every period. The path dependence of held units motivates a date-sequential `numba` kernel (Lam et al., 2015), and array-based analytics operate on its outputs. The backtester suits portfolio research, while order-book events and signal generation remain outside it. Market frictions enter instead as proportional rebalancing costs, set per instrument and per date.

The subpackage `qis.discrete_portfolio` covers strategies that emit orders from a callback on each observation. The replay supports full fills at the next observation and converts the ledgers into the same `PortfolioData` object, reconciled against net asset value. Partial fills, cancellations, and market impact remain outside this version.

We build the chart, the summary table, and the factsheet of a strategy from one `PortfolioData` object, constructed from weights and prices. The same object feeds the three comparisons of a research pipeline: a strategy against its benchmark, a parameter perturbation against its base case, and the sub-strategies of one strategy side by side. One reporting layer renders all three comparisons under the same conventions for performance evaluation.

Beyond backtesting, `qis` supports analysis of currency exposure and factor risk. `FxRatesData` aligns spot exchange rates and short interest rates to express asset returns in a reference currency and evaluate partial currency hedges, including carry implied by covered interest parity. Exponentially weighted factor models estimate time-varying loadings, while risk analytics report portfolio factor exposures and systematic and residual risk contributions. Stress-testing functions apply specified or conditionally derived factor shocks to current holdings and report scenario profit and loss with reconciled factor attribution.

The public interface is `qis.__all__`, which holds 480 names. The module `src/qis/api.py` records that list as a literal, together with a documented core of 166 symbols in 14 capability groups. The broad namespace preserves convenient imports and compatibility for existing

consumers, at the cost of discoverability. The curated core gives new users a smaller entry point without removing any export.

Tests check exported plots on synthetic panels, the documented arguments of the core, local documentation links, and the recorded software measurements this paper quotes. Selected offline examples execute in the suite, and static checks cover the imports and calls of all examples. Plot smoke tests establish that figures draw, while layout and readability require visual review. The suite runs without network access on a core installation, using a frozen, seeded simulator that reproduces the defects of real price panels.

Research impact statement

Within our research stack, `qis` carries the analytics behind our papers, so others can reproduce their calculations with public code. The package supports papers on cryptocurrency allocation (Sepp, 2023) and robust strategic and tactical asset allocation (Sepp et al., 2026). Its bootstrap resampling supports research on capital market assumptions using multi-asset tradable factors, forthcoming in The Journal of Portfolio Management (Sepp et al., 2027). Its regime-conditional analytics support the working paper on trend-following systems (Sepp & Lucic, 2026) and research on portfolio overlays, forthcoming in the Journal of Investment Management (Sepp & Kastenholz, 2026). The [public companion example](#) of the overlay paper demonstrates these calculations on synthetic data.

Two public projects outside our package stack use `qis`. The [example in tradingstrategy-ai/optimalportfolios-defi](#) runs the `qis` backtester, strategy factsheet and report export on an optimised ETF portfolio. The [backtesting example](#) in `luna-007/extra` uses `qis` to compare leveraged ETF allocations with funding costs and to generate a performance factsheet.

AI usage disclosure

From July to October 2026, Anthropic's Claude (Opus 5 and 5.5, Fable 5 and 5.1) and OpenAI Codex (Sol 5.6) assisted with software development, tests, documentation, and paper preparation. The author made the methodological and architectural decisions, reviewed and validated the assisted outputs, and remains responsible for the work. Assisted changes pass through the same test suite and continuous integration as all other changes. The suite checks recorded software measurements against recomputed values or recorded provenance.

Acknowledgements

We thank John D. Reynolds for contributions to numerical correctness, performance, reporting, and regression test coverage in `qis`. This work received no specific funding.

References

- backtrader contributors. (2026). *backtrader: strategy*. <https://www.backtrader.com/docu/strategy/>
- Boyd, S., Busseti, E., Diamond, S., Kahn, R. N., Koh, K., Nystrup, P., & Speth, J. (2017). Multi-period trading via convex optimization. *Foundations and Trends in Optimization*, 3(1), 1–76. <https://doi.org/10.1561/24000000023>
- bt contributors. (2026). *bt: Flexible backtesting for Python*. <https://pmorisette.github.io/bt/>
- empyrial-reloaded contributors. (2026). *empyrial-reloaded: Common financial risk and performance metrics*. <https://empyrial.ml4trading.io/>

- ffn contributors. (2026). *ffn: Financial function library for Python*. <https://pmorisette.github.io/ffn/>
- Geltner, D. (1993). Estimating market values from appraised values without assuming an efficient market. *Journal of Real Estate Research*, 8(3), 325–345. <https://doi.org/10.1080/10835547.1993.12090713>
- Getmansky, M., Lo, A. W., & Makarov, I. (2004). An econometric model of serial correlation and illiquidity in hedge fund returns. *Journal of Financial Economics*, 74(3), 529–609. <https://doi.org/10.1016/j.jfineco.2004.04.001>
- 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., Fernández del Río, J., 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>
- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A LLVM-based Python JIT compiler. *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 1–6. <https://doi.org/10.1145/2833157.2833162>
- Martin, R. A. (2021). PyPortfolioOpt: Portfolio optimization in Python. *Journal of Open Source Software*, 6(61), 3066. <https://doi.org/10.21105/joss.03066>
- McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in science conference* (pp. 56–61). <https://doi.org/10.25080/Majora-92bf1922-00a>
- Nicolini, C., Manzi, M., & Delatte, H. (2026). skfolio: Portfolio optimization in Python. *Proceedings of the 2025 International Research and Industry Symposium on Artificial Intelligence*, 35–41. <https://doi.org/10.65236/iris-ai/proceedings/2025/article/3>
- Peterson, B. G., & Carl, P. (2026). *PerformanceAnalytics: Econometric tools for performance and risk analysis*. <https://doi.org/10.32614/CRAN.package.PerformanceAnalytics>
- Politis, D. N., & Romano, J. P. (1994). The stationary bootstrap. *Journal of the American Statistical Association*, 89(428), 1303–1313. <https://doi.org/10.1080/01621459.1994.10476870>
- pyfolio-reloaded contributors. (2026). *pyfolio-reloaded: API reference*. <https://pyfolio.ml4trading.io/api-reference.html>
- QuantStats contributors. (2026). *QuantStats: Portfolio analytics for quants*. <https://github.com/ranaroussi/quantstats>
- Riskfolio-Lib contributors. (2026). *Riskfolio-Lib: Portfolio optimization in Python*. <https://riskfolio-lib.readthedocs.io/en/latest/>
- Schmelzer, T. (2026). *cvxsimulator: Simple simulator for investors*. <https://github.com/cvxgrp/simulator>
- Seabold, S., & Perktold, J. (2010). statsmodels: Econometric and statistical modeling with Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in science conference* (pp. 92–96). <https://doi.org/10.25080/Majora-92bf1922-011>
- Sepp, A. (2023). Optimal allocation to cryptocurrencies in diversified portfolios. *Risk*, 1–6. <https://doi.org/10.2139/ssrn.4217841>
- Sepp, A., & Dézeraud, L. (2019). Trend-following CTAs vs alternative risk-premia: Crisis beta vs risk-premia alpha. *The Hedge Fund Journal*, 138, 20–31. <https://thehedgerefundjournal.com>

- [com/trend-following-ctas-vs-alternative-risk-premia/](#)
- Sepp, A., Hansen, E. H., & Kastenholz, M. (2027). Capital market assumptions and strategic asset allocation using multi-asset tradable factors. *The Journal of Portfolio Management*. <https://doi.org/10.2139/ssrn.6785958>
- Sepp, A., & Kastenholz, M. (2026). The convexity premium of portfolio overlays. *Journal of Investment Management*.
- Sepp, A., & Lucic, V. (2026). *The science and practice of trend-following systems* [Working paper]. <https://doi.org/10.48550/arXiv.2607.19497>
- Sepp, A., Ossa, I., & Kastenholz, M. (2026). Robust optimization of strategic and tactical asset allocation for multi-asset portfolios. *The Journal of Portfolio Management*, 52(4), 86–120. <https://doi.org/10.3905/jpm.2025.1.806>
- Sheppard, K., & others. (2026). *arch: StationaryBootstrap*. <https://doi.org/10.5281/zenodo.593254>
- vectorbt contributors. (2026). *vectorbt: Portfolio API reference*. <https://vectorbt.dev/api/portfolio/base/>
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261–272. <https://doi.org/10.1038/s41592-019-0686-2>
- Waskom, M. L. (2021). *seaborn: Statistical data visualization*. *Journal of Open Source Software*, 6(60), 3021. <https://doi.org/10.21105/joss.03021>
- zipline-reloaded contributors. (2026). *zipline-reloaded: API reference*. <https://zipline.ml4trading.io/api-reference.html>