Empyrial

Homepage

What is Empyrial?

Empyrial is a Python-based **open-source quantitative investment** library dedicated to **financial institutions** and **retail investors**, officially released in Mars 2021. Already used by **thousands of people working in the finance industry**, Empyrial aims to become an all-in-one platform for **portfolio management**, **analysis**, and **optimization**.

Empyrial **empowers portfolio management** by bringing the best of **performance and risk analysis** in an **easy-to-understand**, **flexible** and **powerful framework**.

With Empyrial, you can easily analyze security or a portfolio in order to **get the best insights from it**.

Installation

You can install Empyrial using pip:

```
pip install empyrial
```

For a better experience, **we advise you to use Empyrial on a notebook** (Jupyter, Google Colab...)

Note: macOS users will need to install Command Line Tools.

Note: if you are on windows, you first need to install C++. (download, install instructions)

Quickstart

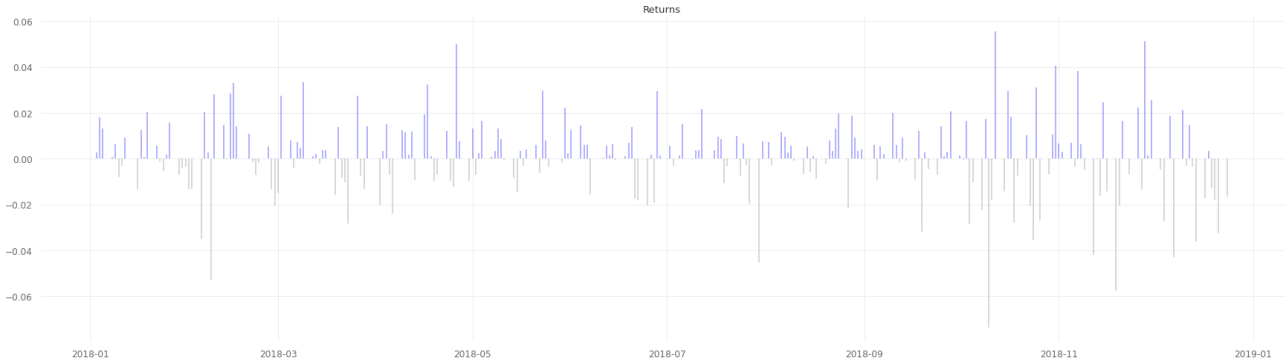
```
from empyrial import empyrial, Engine

portfolio = Engine(
                    start_date= "2018-06-09", #start date for the backtestin
                    portfolio= ["BABA", "PDD", "KO", "AMD","^IXIC"], #assets
                    weights = [0.2, 0.2, 0.2, 0.2, 0.2], #equal weighting is
                    benchmark = ["SPY"] #SPY is set by default
)

empyrial(portfolio)
```

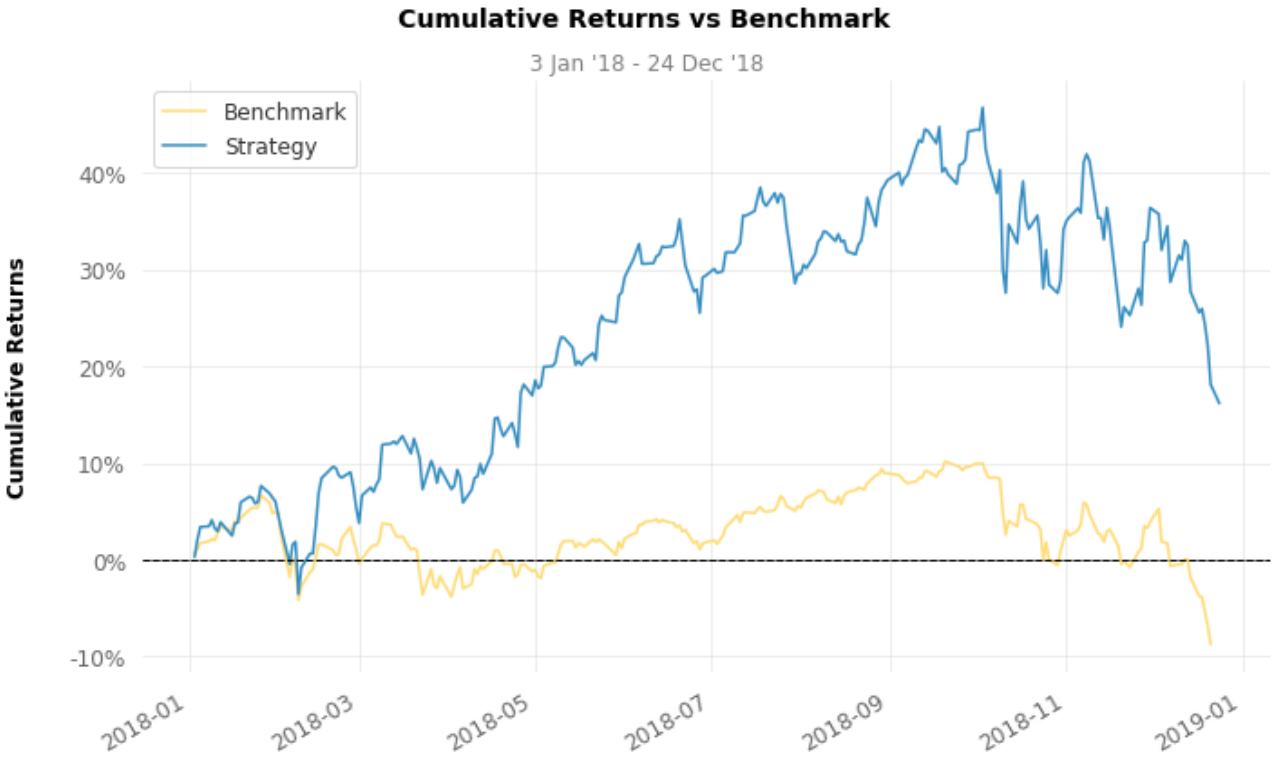
Output

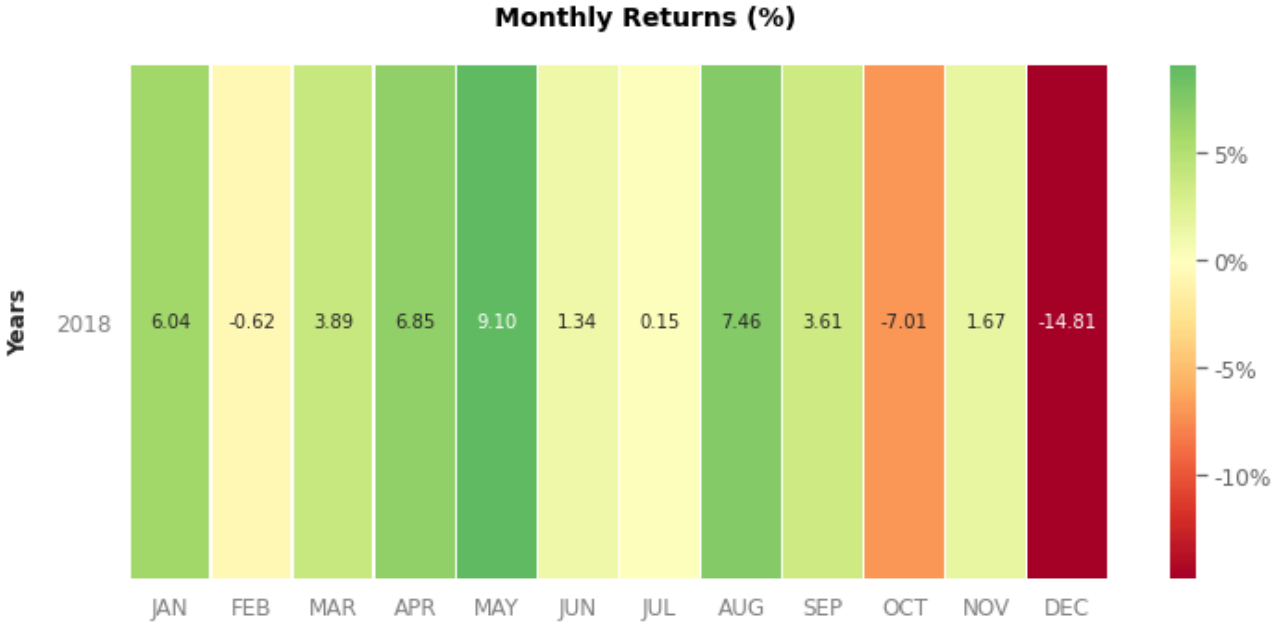
Start date: 2018-01-01
End date: 2018-12-24

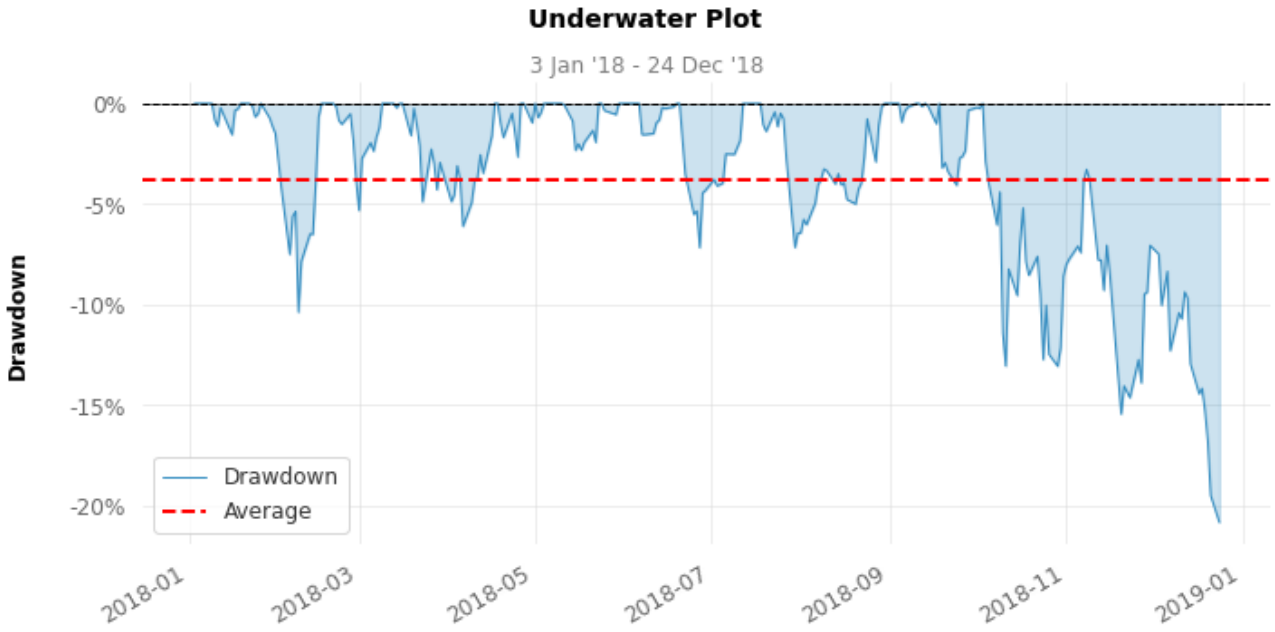
Backtest

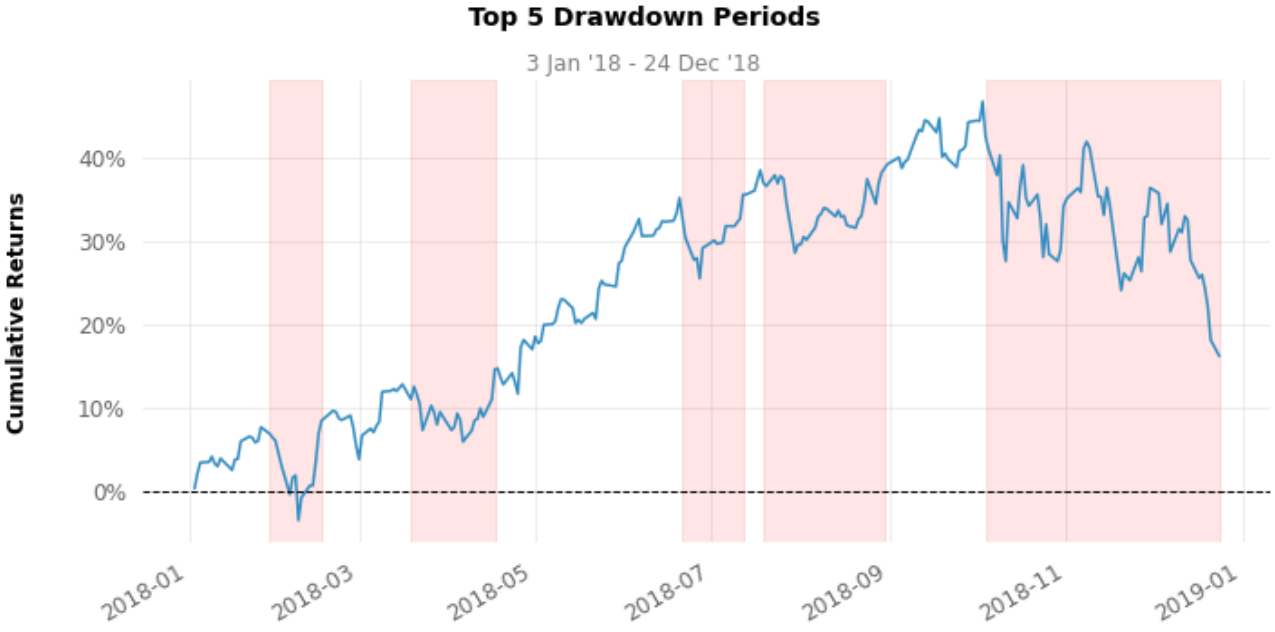
Annual return	17.0%
Cumulative return	16.15%
Annual volatility	27.59 %
Winning day ratio	56.1%
Sharpe ratio	0.69
Calmar ratio	0.8
Information ratio	0.01
Stability	0.69
Max Drawdown	-20.83 %
Sortino ratio	0.97
Skew	-0.44
Kurtosis	2.21
Tail Ratio	1.0
Common sense ratio	1.13
Daily value at risk	-3.0 %
Alpha	0.37
Beta	1.3

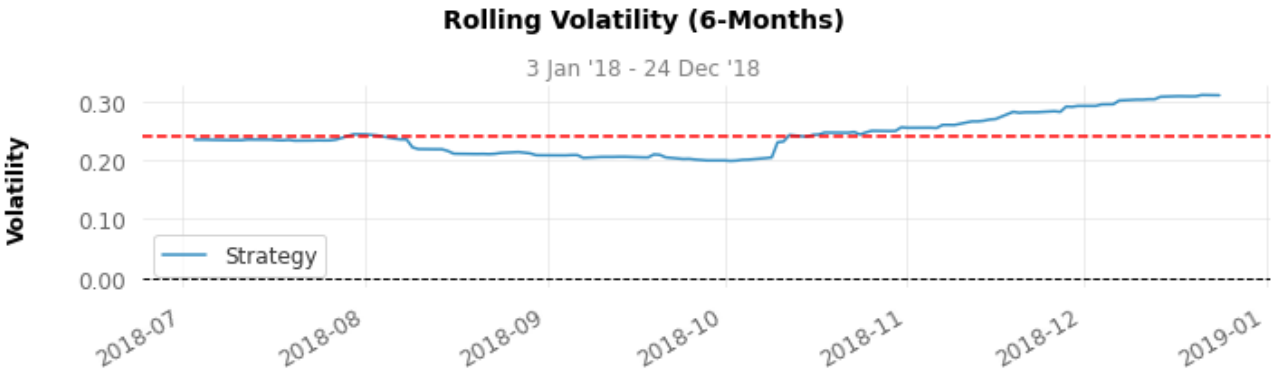


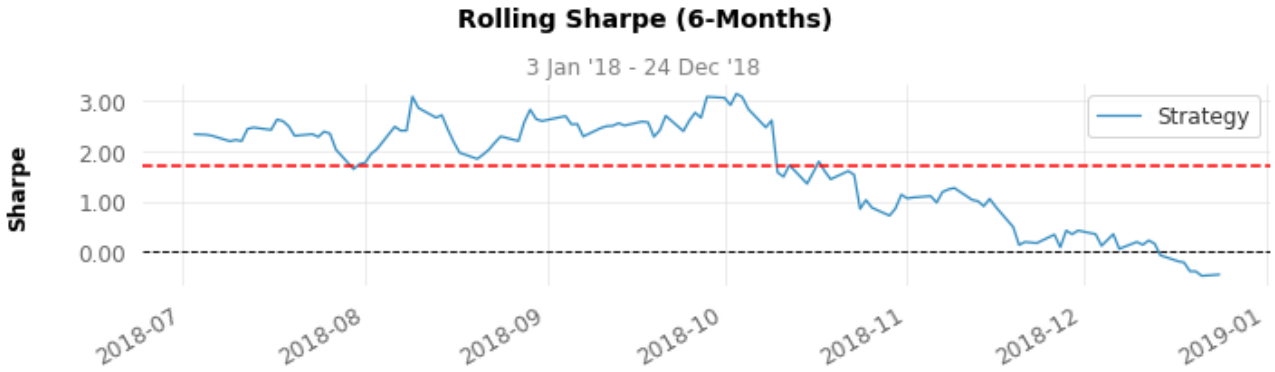

Cumulative Returns vs Benchmark
3 Jan '18 - 24 Dec '18
Benchmark
Strategy
Cumulative Returns
40%
30%
20%
10%
0%
-10%
2018-01
2018-03
2018-05
2018-07
2018-09
2018-11
2019-01

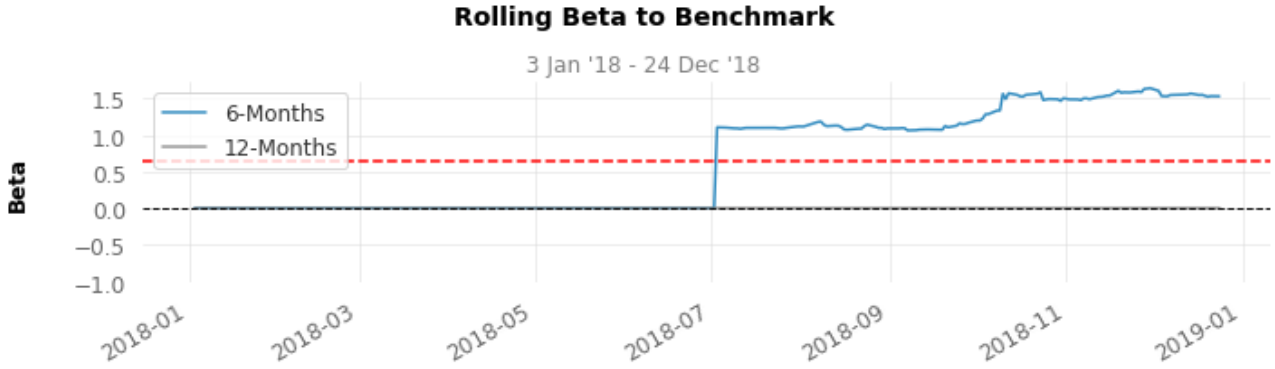

Monthly Returns (%)
Years
2018
6.04
-0.62
3.89
6.85
9.10
1.34
0.15
7.46
3.61
-7.01
1.67
-14.81
JAN
FEB
MAR
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
5%
0%
-5%
-10%

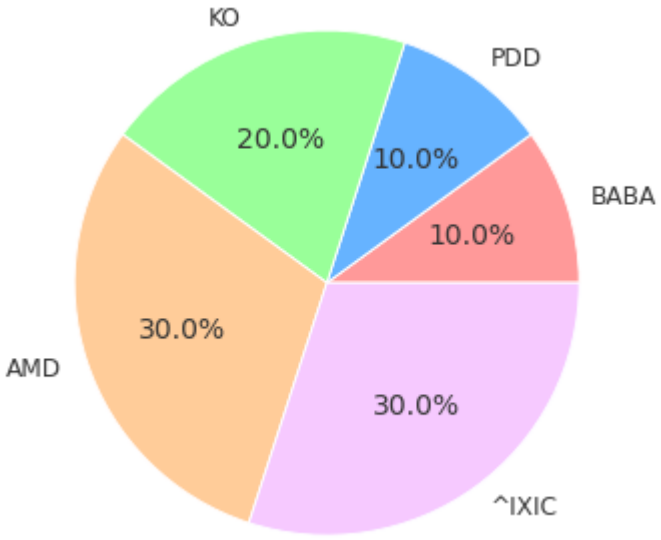

Underwater Plot
3 Jan '18 - 24 Dec '18
Drawdown
0%
-5%
-10%
-15%
-20%
Drawdown
Average
2018-01
2018-03
2018-05
2018-07
2018-09
2018-11
2019-01


Top 5 Drawdown Periods
3 Jan '18 - 24 Dec '18
Cumulative Returns
40%
30%
20%
10%
0%
2018-01
2018-03
2018-05
2018-07
2018-09
2018-11
2019-01


Rolling Volatility (6-Months)
3 Jan '18 - 24 Dec '18
Volatility
0.30
0.20
0.10
0.00
Strategy
2018-07
2018-08
2018-09
2018-10
2018-11
2018-12
2019-01


Rolling Sharpe (6-Months)
3 Jan '18 - 24 Dec '18
Sharpe
3.00
2.00
1.00
0.00
Strategy
2018-07
2018-08
2018-09
2018-10
2018-11
2018-12
2019-01


Rolling Beta to Benchmark
3 Jan '18 - 24 Dec '18
Beta
1.5
1.0
0.5
0.0
−0.5
−1.0
6-Months
12-Months
2018-01
2018-03
2018-05
2018-07
2018-09
2018-11
2019-01


KO
PDD
BABA
AMD
^IXIC
20.0%
10.0%
10.0%
30.0%
30.0%

Engine

Engine's argument:

- `start_date` : start date for the backtesting (format: YYYY/MM/DD)
- `end_date` : end date for the backtesting. The end date is by default today's date. (format: YYYY/MM/DD)
- `portfolio` : assets you invest in (tickers in a list)
- `weights` : allocation of the capital in every asset (proportion in a list). The default allocation is equal weighting.
- `benchmark` : the benchmark for the backtesting. SPY is the default benchmark.
- `optimizer` : portfolio optimizer used to allocate capital in your strategy.
- `rebalance` : rebalance frequency (for calendar rebalancing) or custom rebalance dates (in a list, format: YYYY/MM/DD)
- `max_vol` : max level of volatility for the Mean-Variance ("MEANVAR") optimizer only. The default value is 0.15.
- `diversification` : level of diversification in the allocation (works with every optimizer except Efficient Frontier, "EF"). The default value is 1.
- `max_weights` : maximum proportion of capital you can invest in a single asset.
- `min_weights` : minimum proportion of capital you can invest in a single asset.
- `risk_manager` : risk management for your strategy (Stop Loss, Take Profit, Max Drawdown).

Empyrial

`empyrial` (*my_portfolio, rf=0.0, confidence_value=0.95*)

- `my_portfolio` : the portfolio you define through Engine.
- `rf` : risk-free rate
- `confidence_value` : confidence proportion for the value-at-risk.

Rebalancing

Calendar Rebalancing

Calendar rebalancing is the most rudimentary rebalancing approach. This strategy simply involves analyzing the investment holdings within the portfolio at predetermined time intervals and adjusting to the original allocation at the desired frequency. (Source: Investopedia)

```
from empyrial import empyrial, Engine

portfolio = Engine(
        start_date= "2018-06-09",
        portfolio= ["BABA", "PDD", "KO", "AMD","^IXIC"],
        weights = [0.2, 0.2, 0.2, 0.2, 0.2], #equal weighting is set by def
        benchmark = ["SPY"], #SPY is set by default
        rebalance = "1y" #rebalance every year
)

empyrial(portfolio)
```

Time periods available for rebalancing are `2y` , `1y` , `6mo` , `quarterly` , `monthly`

Custom Rebalancing

You can decide custom dates for rebalancing, by doing this:

```
from empyrial import empyrial, Engine

portfolio = Engine(
        start_date= "2018-06-09",
        portfolio= ["BABA", "PDD", "KO", "AMD","^IXIC"],
        weights = [0.2, 0.2, 0.2, 0.2, 0.2], #equal weighting by default
        benchmark = ["SPY"], #SPY by default
        rebalance = ["2018-06-09", "2019-01-01", "2020-01-01", "2021-01-01
)

empyrial(portfolio)
```

⚠ In that case, make sure, that the 1st element of the list corresponds to the `start_date` and the last element corresponds to the `end_date` which is **today's date** by default.

Optimization

Global Efficient Frontier

The efficient frontier is the **set of optimal portfolios** that offer the **highest expected return for a defined level of risk** or the lowest risk for a given level of expected return. (Source: Investopedia)

```
from empyrial import empyrial, Engine

portfolio = Engine(
      start_date = "2018-01-01",
      benchmark = ["SPY"], #SPY is set by default
      portfolio = ["BABA", "PDD", "KO", "AMD","^IXIC"],
      optimizer = "EF"
)

empyrial(portfolio)
```

When using this optimizer, the weights may be not well-diversified, you can fix that by reading the "Diversification" section :

→ Diversification /optimization/diversification

Mean-Variance

Mean-variance analysis is the process of weighing the risk, expressed as variance, against expected return. Investors weigh how much risk they are willing to take on in exchange for different levels of reward. The mean-variance analysis allows investors to **find the biggest reward at a given level of risk**. (Source: Investopedia)

```
from empyrial import empyrial, Engine

portfolio = Engine(
        start_date= "2020-06-09",
        benchmark = ["SPY"], #SPY is set by default
        portfolio= ["BABA", "AAPL", "KO", "^DJI","^IXIC"],
        optimizer = "MEANVAR", # defines Mean-Variance as the optimizer
        max_vol = 0.25, #maximize the return for this level of volatility (
)

empyrial(portfolio)
```

⚠ If the `max_vol` value is too low or if the assets in `portfolio` are too volatile, it might give you an error.

Hierarchical Risk Parity

Hierarchical Risk Parity (HRP) is a **risk-based portfolio optimization** algorithm, which has been shown to generate **diversified portfolios** with robust out-of-sample properties without the need for a positive-definite return covariance matrix (Source: https://ideas.repec.org/p/sza/wpaper/wpapers328.html)

```
from empyrial import empyrial, Engine

portfolio = Engine(
      start_date = "2018-01-01",
      benchmark = ["SPY"], #SPY is set by default
      portfolio = ["BABA", "PDD", "KO", "AMD","^IXIC"],
      optimizer = "HRP"
)

empyrial(portfolio)
```

Minimum-Variance

A **Minimum Variance** portfolio is a collection of securities that combine to **minimize the price volatility** of the overall portfolio. (Source: the balance)

```
from empyrial import empyrial, Engine

portfolio = Engine(
      start_date = "2018-01-01",
      benchmark = ["SPY"], #SPY is set by default
      portfolio = ["BABA", "PDD", "KO", "AMD","^IXIC"],
      optimizer = "MINVAR"
)

empyrial(portfolio)
```

Diversification

If you want to diversify more or less your portfolio, there are 2 ways to do it with Empyrial:

- By using `min_weights` and `max_weights` :

Let's take this example:

```
from empyrial import empyrial, Engine

portfolio = Engine(
      start_date = "2018-01-01",
      benchmark = ["SPY"], #SPY is set by default
      portfolio = ["BABA", "PDD", "KO", "AMD","^IXIC"],
      optimizer = "EF"
)

empyrial(portfolio)
```

```
portfolio.weights
```

Output

```
[0.0, 0.0, 0.02258, 0.97742, 0.0]
```

As you can see the allocation is very clustered around AMD and this is a problem. To solve that, we can do:

```
from empyrial import empyrial, Engine

portfolio = Engine(
      start_date = "2018-01-01",
      benchmark = ["SPY"], #SPY is set by default
      portfolio = ["BABA", "PDD", "KO", "AMD","^IXIC"],
      optimizer = "EF",
      min_weights = 0.05, #invest at least 5% of the capital in every asse
```

```
        max_weights = 0.35 #don't invest more than 35% in one asset
)

empyrial(portfolio)
```

```
portfolio.weights
```

Output

```
[0.05, 0.05, 0.2, 0.35, 0.35]
```

So, we can tune these two parameters (`min_weights` and `max_weights`) in order to get a better allocation.

- The second way is by using `diversification` (works with every optimizer except the Efficient Frontier, "EF"):

`diversification` 's default value is 1.

The higher is this value, the more it diversifies the portfolio and gets closer to equal weighting.

The lower is this value, the less it diversifies the portfolio.

Example:

```
from empyrial import empyrial, Engine

portfolio = Engine(
        start_date = "2018-01-01",
        benchmark = ["SPY"],
        portfolio = ["BABA", "PDD", "KO", "AMD","^IXIC"],
        optimizer = "MINVAR",
        diversification = 1.8
)

empyrial(portfolio)
```

Custom allocation

You can use custom weights:

```
from empyrial import empyrial, Engine

portfolio = Engine(
      start_date = "2018-01-01",
      portfolio= ["BABA", "PDD", "KO", "AMD","^IXIC"],
      weights = [0.1, 0.3, 0.15, 0.25, 0.2], #custom weights
)

empyrial(portfolio)
```

Risk Management

Max Drawdown

A **maximum drawdown** (MDD) is the **maximum observed loss from a peak** to a trough of a portfolio before a new peak is attained. Maximum drawdown is an indicator of **downside risk** over a specified time period. (Source: Investopedia)

```
from empyrial import empyrial, Engine

portfolio = Engine(
      start_date = "2018-01-01",
      portfolio= ["BABA", "PDD", "KO", "AMD","^IXIC"],
      risk_manager = {"Max Drawdown" : -0.2} #Stop the investment when the
)

empyrial(portfolio)
```

Take Profit

A **take-profit order** (T/P) is a type of limit order that specifies the **exact price at which to close out** an open position for a profit. If the price of the security does not reach the limit price, the take-profit order does not get filled. (Source: Investopedia)

```
from empyrial import empyrial, Engine

portfolio = Engine(
      start_date = "2018-01-01",
      portfolio= ["BABA", "PDD", "KO", "AMD","^IXIC"],
      optimizer = "EF",
      rebalance = "1y", #rebalance every year
      risk_manager = {"Take Profit" : 0.25} #Stop the investment when the
)

empyrial(portfolio)
```

Stop Loss

A **stop-loss** is designed to **limit an investor's loss** on a security position. For example, setting a stop-loss order for 10% below the price at which you bought the stock will limit your loss to 10%. (Source: Investopedia)

```
from empyrial import empyrial, Engine

portfolio = Engine(
      start_date = "2018-01-01",
      portfolio= ["BABA", "PDD", "KO", "AMD","^IXIC"],
      optimizer = "EF",
      rebalance = "1y", #rebalance every year
      risk_manager = {"Stop Loss" : -0.2} #Stop the investment when the lo
)

empyrial(portfolio)
```

Orderbook

Access the orderbook

First, run the backtesting.

```
from empyrial import empyrial, Engine

portfolio = Engine(
      start_date = "2018-01-01",
      portfolio = ["BLK", "BAC", "AAPL", "TM", "JPM","JD", "INTU", "NVDA",
      optimizer = "EF",
      rebalance = "1y", #rebalance every year
)

empyrial(portfolio)
```

Then, run:

```
empyrial.orderbook
```

Output

	2019-01-01	2020-01-01	2020-12-31	2021-07-27
BLK	0.05	0.05000	0.05	0.05000
BAC	0.05	0.05000	0.05	0.05000
AAPL	0.05	0.33845	0.20	0.20479
TM	0.05	0.05000	0.05	0.05000
JPM	0.05	0.05000	0.05	0.05000
JD	0.05	0.05000	0.05	0.05000
INTU	0.40	0.26155	0.05	0.10126
NVDA	0.05	0.05000	0.05	0.05000
DIS	0.05	0.05000	0.05	0.05000
TSLA	0.20	0.05000	0.40	0.34395

Save the tearsheet

Get a report

If you want to get a **PDF report** of the backtesting use `get_report` instead of `empyrial` function:

```
from empyrial import get_report, Engine

portfolio = Engine(
      start_date = "2018-01-01",
      portfolio = ["BLK", "BAC", "AAPL", "TM", "JPM","JD", "INTU", "NVDA",
      optimizer = "EF",
      rebalance = "1y",
      risk_manager = {"Stop Loss" : -0.2}
)

get_report(portfolio)
```

Output