Strategy return analysis

FINANCIAL TRADING IN PYTHON

Chelsea Yang
Data Science Instructor

datacamp

Obtain detailed backtest stats

```
# Get all backtest stats
resInfo = bt_result.stats
print(resInfo.index)
```

```
Index(['start', 'end', 'rf', 'total_return', 'cagr', 'max_drawdown', 'calmar',
       'mtd', 'three_month', 'six_month', 'ytd', 'one_year', 'three_year',
       'five_year', 'ten_year', 'incep', 'daily_sharpe', 'daily_sortino',
       'daily_mean', 'daily_vol', 'daily_skew', 'daily_kurt', 'best_day',
       'worst_day', 'monthly_sharpe', 'monthly_sortino', 'monthly_mean',
       'monthly_vol', 'monthly_skew', 'monthly_kurt', 'best_month',
       'worst_month', 'yearly_sharpe', 'yearly_sortino', 'yearly_mean',
       'yearly_vol', 'yearly_skew', 'yearly_kurt', 'best_year', 'worst_year',
       'avg_drawdown', 'avg_drawdown_days', 'avg_up_month', 'avg_down_month',
       'win_year_perc', 'twelve_month_win_perc'],
      dtype='object')
```

Strategy returns

$$Return = (V_e - V_b)/V_b$$

V_e: ending value

V_v: beginning value

```
# Get daily, monthly and yearly returns
print('Daily return: %.4f'% resInfo.loc['daily_mean'])
print('Monthly return: %.4f'% resInfo.loc['monthly_mean'])
print('Yearly return: %.4f'% resInfo.loc['yearly_mean'])
```

```
Daily return: 0.1966
Monthly return: 0.2207
Yearly return: 0.3328
```

Compound annual growth rate

$$CAGR = (V_f / V_i)^{\frac{1}{n}} - 1$$

V_f: final value

V_i: initial value

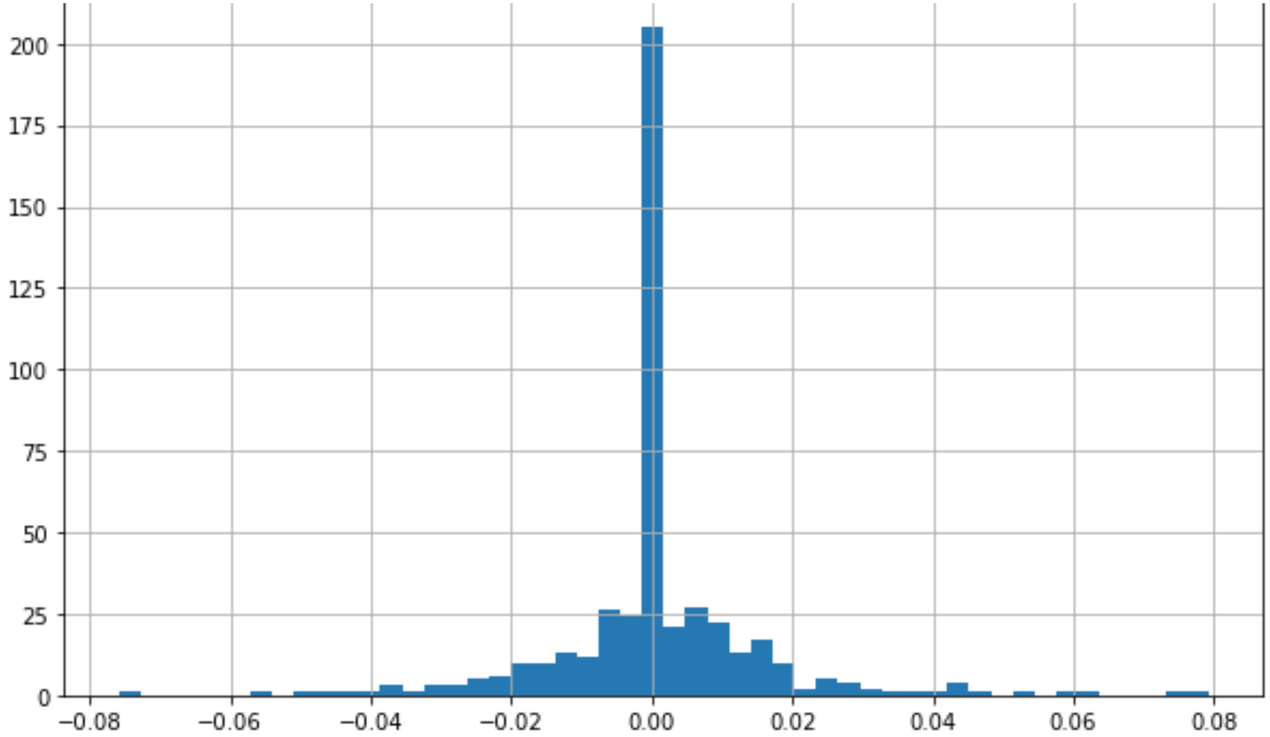
n: number of years

```
# Get the compound annual growth rate
print('Compound annual growth rate: %.4f'% resInfo.loc['cagr'])
```

```
Compound annual growth rate: 0.1855
```

Plot return histogram

```
# Plot the weekly return histogram
bt_result.plot_histograms(bins=50, freq = 'w')
```



Compare strategy lookback returns

```
# Get the lookback returns
lookback_returns = bt_result.display_lookback_returns()
print(lookback_returns)
```

```
      Strategy1 Strategy2
mtd        3.30%    -0.03%
3m         0.68%    -2.15%
6m         8.11%     8.32%
ytd       28.08%    10.46%
1y        35.20%    17.09%
3y        11.48%    11.01%
5y         9.35%     9.48%
10y        9.35%     9.48%
incep      9.35%     9.48%
```

Let's practice!

FINANCIAL TRADING IN PYTHON

Drawdown

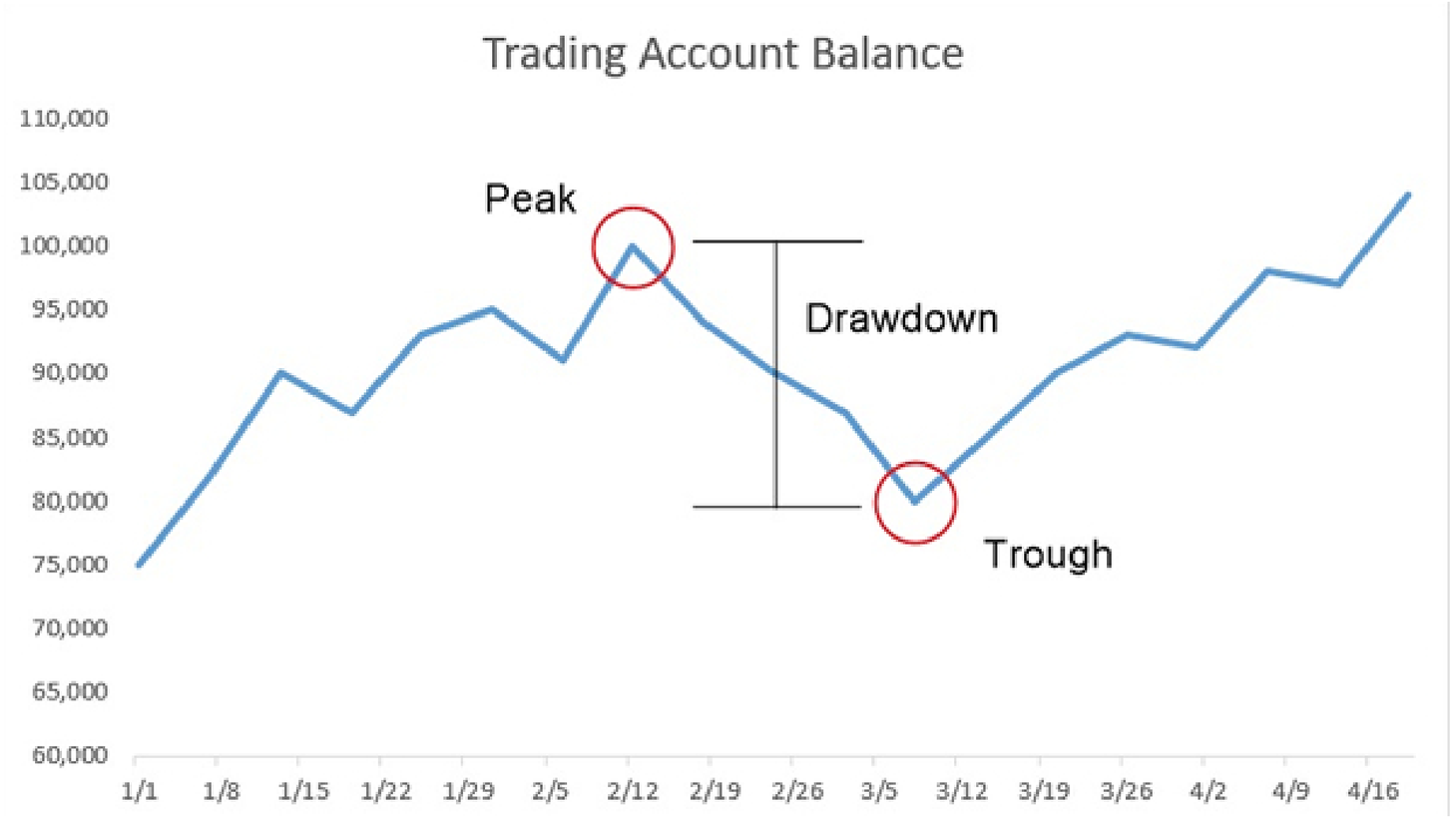
FINANCIAL TRADING IN PYTHON

Chelsea Yang
Data Science Instructor

datacamp

What is a drawdown?

A drawdown is a peak-to-trough decline during a specific period for an asset or a trading account.

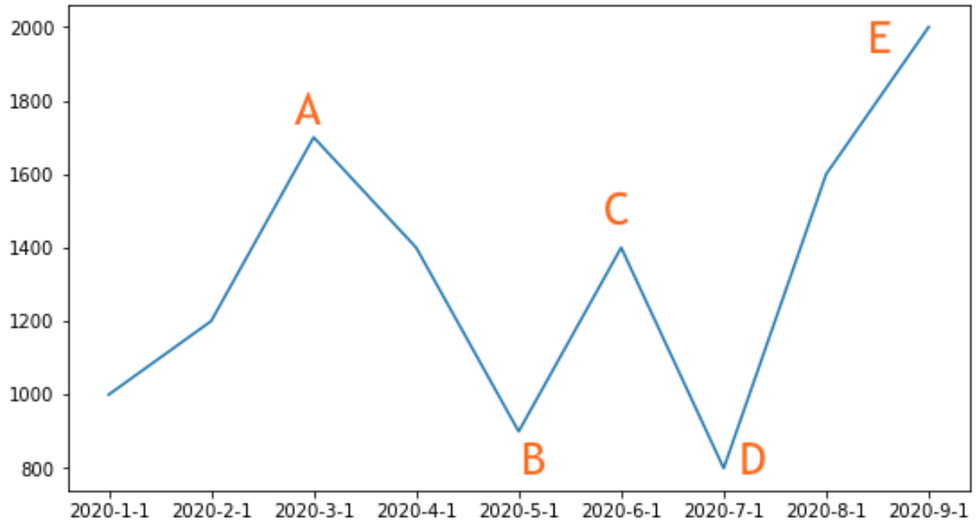


Max drawdown

$$\text{Max Drawdown} = (V_p - V_l)/V_l$$

V_p: Peak value before the largest drop

V_l: Lowest value before a new high value



Max drawdown

= (Point A value - point D value)/Point A value

= (1700 - 800)/1700 = 53%

Obtain drawdowns from backtest stats

```
resInfo = bt_result.stats
# Get the max drawdown
max_drawdown = resInfo.loc['max_drawdown']
print('Maximum drawdown: %.2f'% max_drawdown)
# Get the average drawdown
avg_drawdown = resInfo.loc['avg_drawdown']
print('Average drawdown: %.2f'% avg_drawdown)
# Get the average drawdown days
avg_drawdown_days = resInfo.loc['avg_drawdown_days']
print('Average drawdown days: %.0f'% avg_drawdown_days)
```

```
Maximum drawdown: -0.59
Average drawdown: -0.11
Average drawdown days: 22
```

The Calmar ratio

CALMAR: CALifornia Managed Accounts Report

$$Calmar = CAGR / \text{Max Drawdown}$$

- The higher the Calmar ratio, the better a strategy performed on a risk-adjusted basis.
- Typically a Camlar ratio larger than 3 is considered excellent.

Calculate the Calmar ratio manually

```
resInfo = bt_result.stats
# Get the CAGR
cagr = resInfo.loc['cagr']
# Get the max drawdown
max_drawdown = resInfo.loc['max_drawdown']

# Calculate Calmar ratio mannually
calmar_calc = cagr / max_drawdown * (-1)
print('Calmar Ratio calculated: %.2f'% calmar_calc)
```

```
Calmar Ratio calculated: 4.14
```

Obtain the Calmar ratio from backtest stats

```
resInfo = bt_result.stats

# Get the Calmar ratio
calmar = resInfo.loc['calmar']
print('Calmar Ratio: %.2f'% calmar)
```

```
Calmar Ratio: 4.14
```

Let's practice!

FINANCIAL TRADING IN PYTHON

Sharpe ratio and Sortino ratio

FINANCIAL TRADING IN PYTHON

Chelsea Yang
Data Science Instructor

datacamp

Which strategy performs better?

Strategy 1:

- Return: 15%
- Volatility (standard deviation): 30%

Strategy 2:

- Return: 10%
- Volatility (standard deviation): 8%

Risk-adjusted return

- Make performance comparable among different strategies
- A ratio that describes risk involved in obtaining the return

Sharpe ratio

$$\text{Sharpe Ratio} = (R_p - R_r)/\sigma_p$$

- R_p: Return of a strategy, portfolio, asset, etc.
- R_r: Risk-free rate
- σ_p: Standard deviation of the excess return $(R_p - R_f)$

- The bigger the Sharpe ratio, the more attractive the return

Now choose again

Strategy 1:

- Return: 15%
- Volatility (standard deviation): 30%
- Sharpe ratio: 15%/30% = 0.5

Strategy 2:

- Return: 10%
- Volatility (standard deviation): 8%
- Sharpe ratio: 10%/8% = 1.25

Obtain Sharpe ratio from bt backtest

```
resInfo = bt_result.stats
```

```
# Get Sharpe ratios from the backtest stats
print('Sharpe ratio daily: %.2f'% resInfo.loc['daily_sharpe'])
print('Sharpe ratio monthly %.2f'% resInfo.loc['monthly_sharpe'])
print('Sharpe ratio annually %.2f'% resInfo.loc['yearly_sharpe'])
```

```
Sharpe ratio daily: 0.49
Sharpe ratio monthly 0.48
Sharpe ratio annually 1.34
```

Calculate Sharpe ratio manually

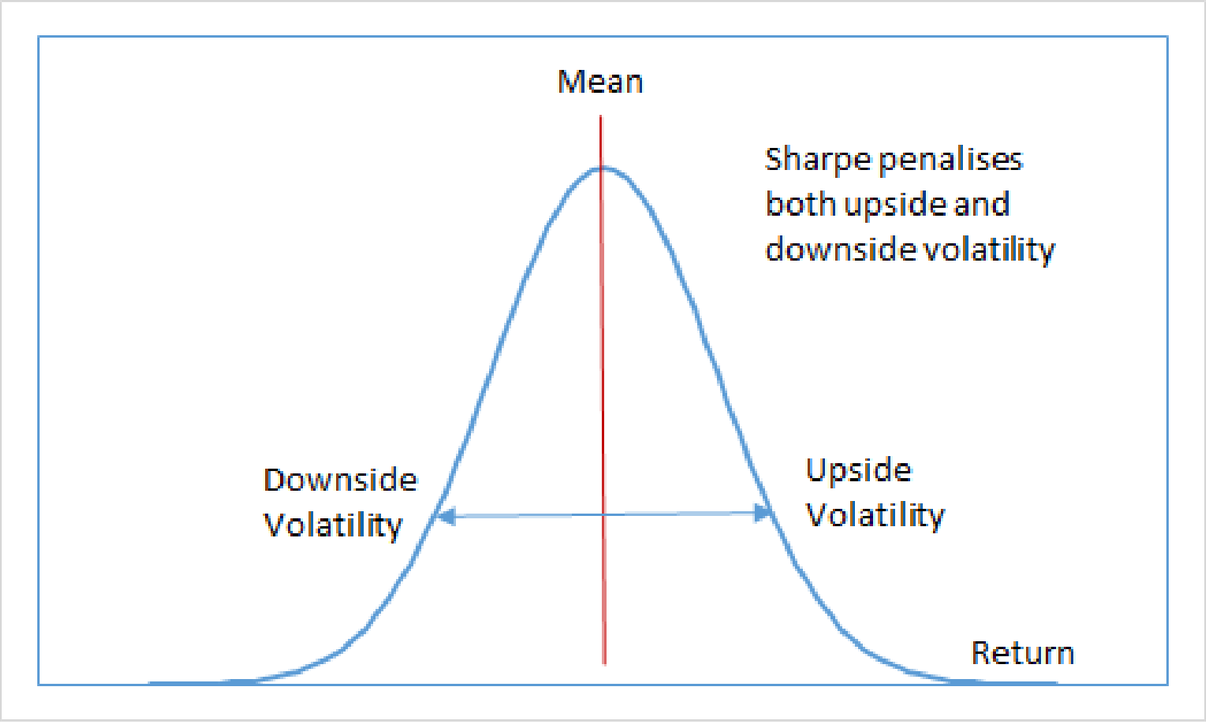
```
# Obtain annual return
annual_return = resInfo.loc['yearly_mean']
# Obtain annual volatility
volatility = resInfo.loc['yearly_vol']

# Calculate Sharpe ratio manually
sharpe_ratio = annual_return / volatility
print('Sharpe ratio annually %.2f'% sharpe_ratio)
```

```
Sharpe ratio annually 1.34
```

Limitations of Sharpe ratio

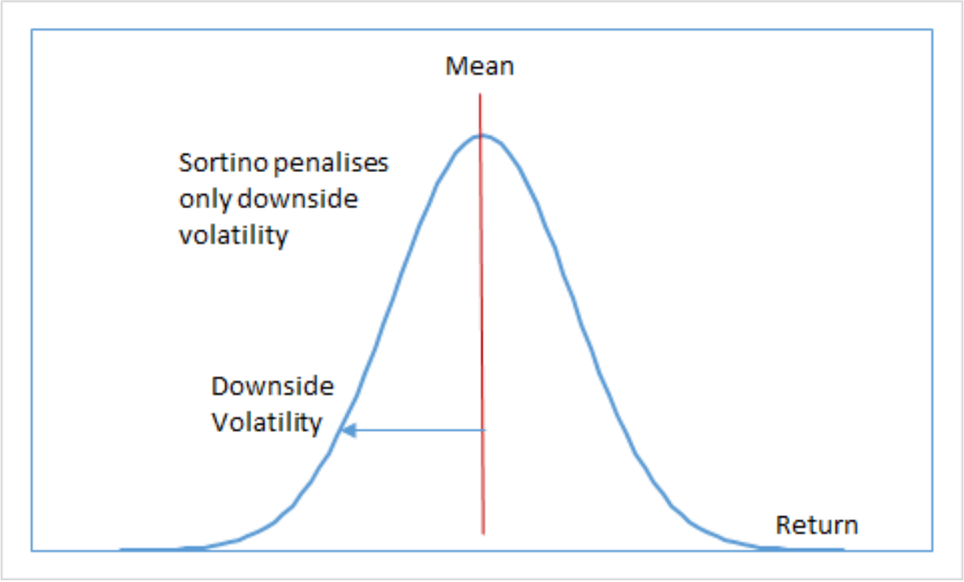
- Penalize both the "good" and "bad" volatility
- Upside volatility can skew the ratio downward



Sortino ratio

$$\text{Sortino Ratio} = (R_p - R_r)/\sigma_d$$

- R_p: Return of a strategy, portfolio, asset, etc
- R_r: Risk-free rate
- σ_d: Downside deviation of the excess return $(R_p - R_f)$



Obtain Sortino ratio from bt backtest

```
resInfo = bt_result.stats
```

```
# Get Sortino ratio from backtest stats
print('Sortino ratio daily: %.2f'% resInfo.loc['daily_sortino'])
print('Sortino ratio monthly %.2f'% resInfo.loc['monthly_sortino'])
print('Sortino ratio annually %.2f'% resInfo.loc['yearly_sortino'])
```

```
Sortino ratio daily: 0.70
Sortino ratio monthly 0.86
Sortino ratio annually 2.29
```

Let's practice!

FINANCIAL TRADING IN PYTHON

Congratulations!

FINANCIAL TRADING IN PYTHON

Chelsea Yang
Data Science Instructor

You did it!

- Get trading data, check and plot the data
- Calculate technical indicators
- Build signal-based strategies and perform strategy backtesting
- Conduct strategy optimization and benchmarking
- Evaluate strategy performance

Thank you!

FINANCIAL TRADING IN PYTHON