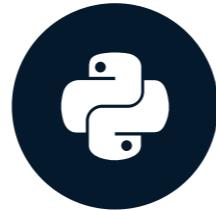


Measuring Risk

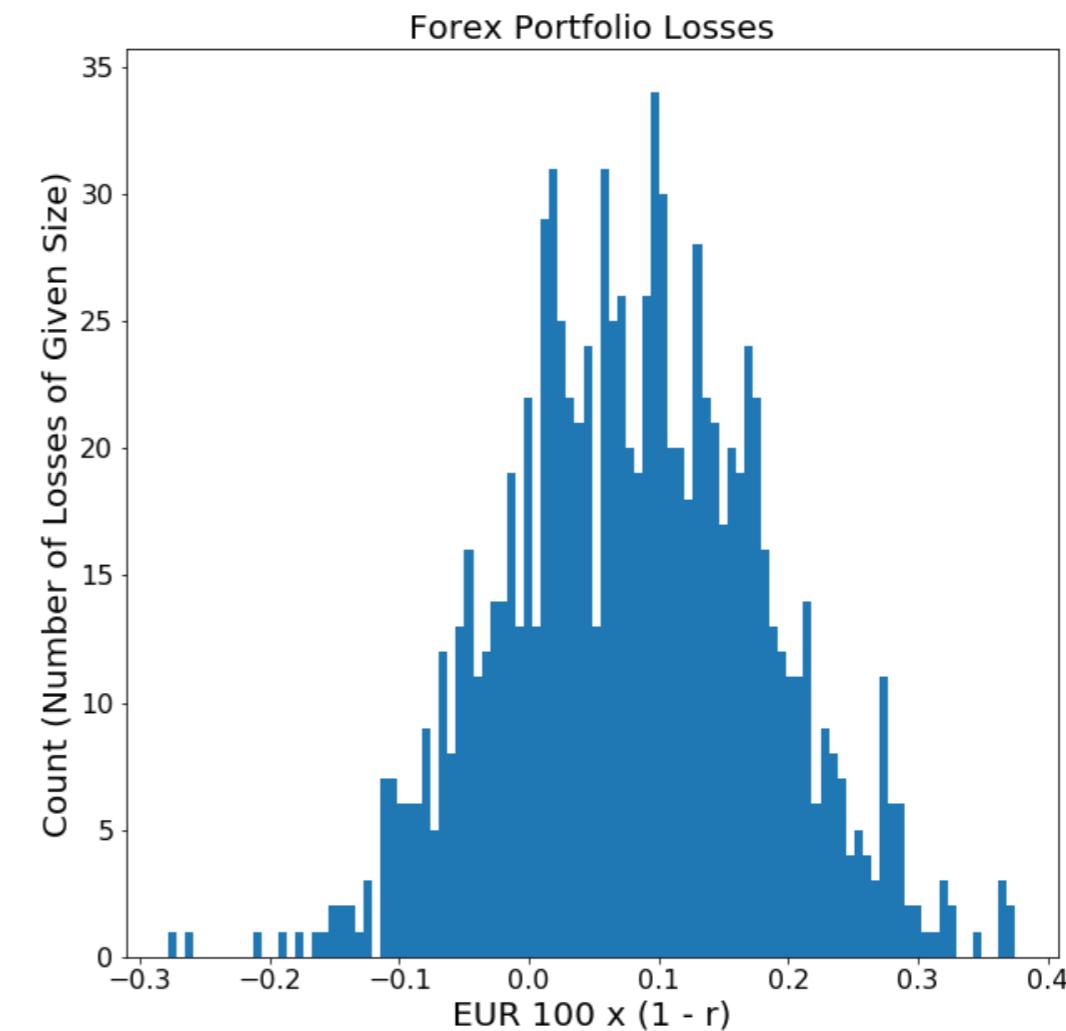
QUANTITATIVE RISK MANAGEMENT IN PYTHON



Jamsheed Shorish
CEO, Shorish Research

The Loss Distribution

- **Forex Example:**
 - Portfolio value in U.S. dollars is USD 100
 - Risk factor =  /  exchange rate
 - Portfolio value in EURO if $1 \text{ } \img{EU}{268} = 1 \text{ } \img{US}{268}$:
USD **100** \times EUR 1 / USD 1 = EUR 100.
 - Portfolio value in EURO if $r \text{ } \img{EU}{268} = 1 \text{ } \img{US}{268}$: =
USD **100** \times EUR r / 1 USD = EUR $100 \times r$
 - **Loss** = EUR 100 - EUR $100 \times r$ = EUR $100 \times (1 - r)$
- **Loss distribution:** Random realizations of r
=> **distribution** of portfolio losses *in the future*

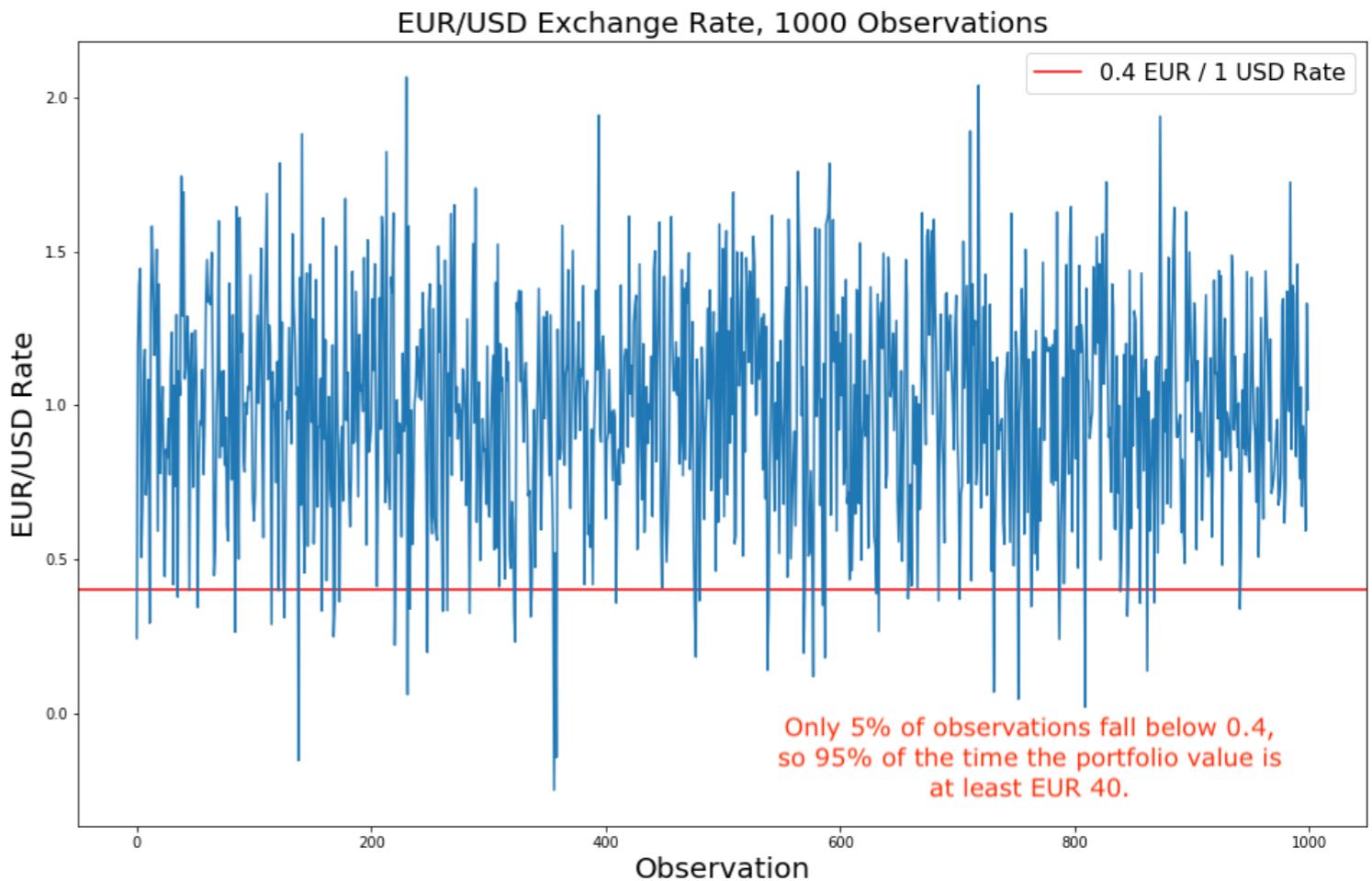


Maximum loss

- What is the maximum loss of a portfolio?
- Losses cannot be bounded with 100% certainty
- **Confidence Level:** replace 100% certainty with *likelihood* of upper bound
- Can express questions like "What is the maximum loss that would take place 95% of the time?"
 - Here the confidence level is **95%**.

Value at Risk (VaR)

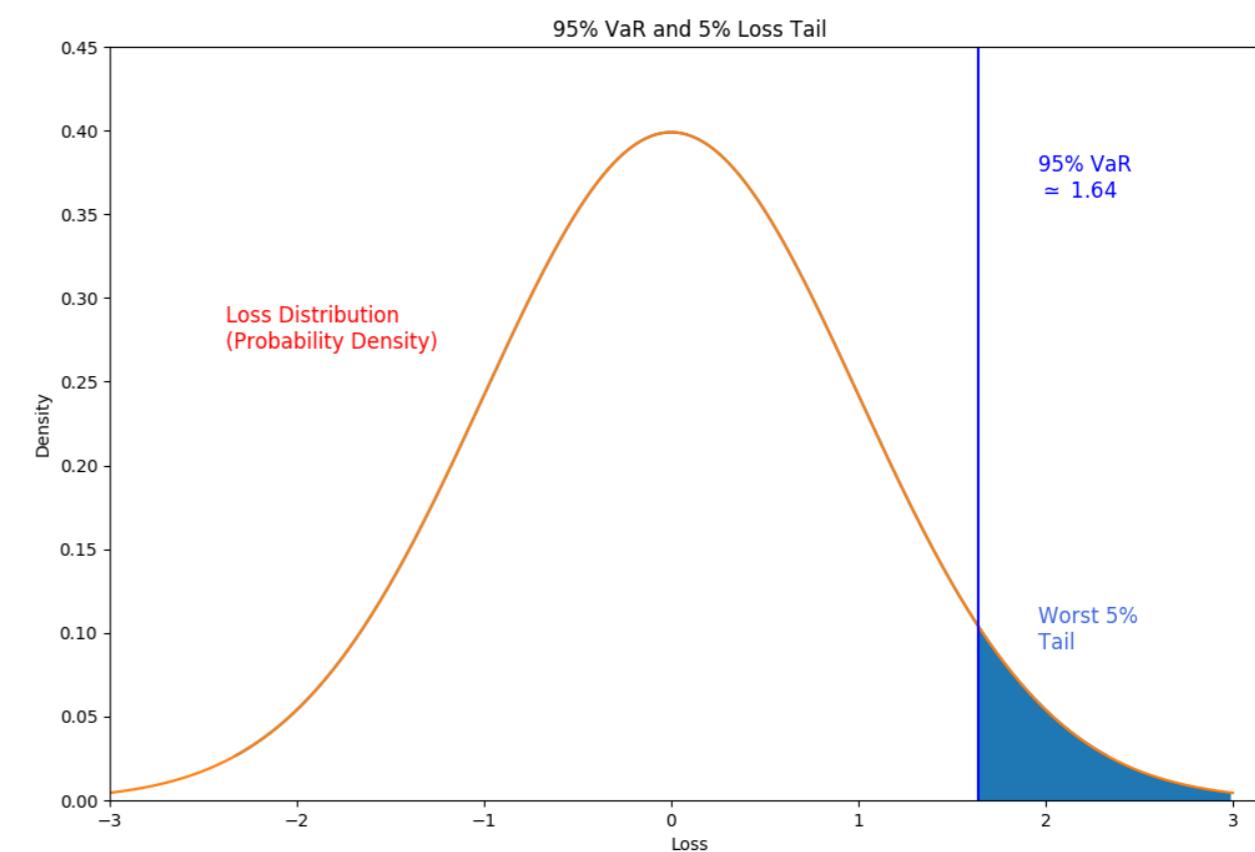
- **VaR**: statistic measuring **maximum portfolio loss** at a particular confidence level
- Typical confidence levels: **95%**, **99%**, and **99.5%** (usually represented as decimals)
- **Forex Example**: If 95% of the time EUR / USD exchange rate is *at least* 0.40, then:
 - portfolio value is *at least* $\text{USD } 100 \times 0.40$
 $\text{EUR} / \text{USD} = \text{EUR } 40$,
 - portfolio loss is *at most* $\text{EUR } 40 - \text{EUR } 100$
 $= \text{EUR } 60$,
 - so the 95% VaR is EUR 60.



Conditional Value at Risk (CVaR)

- CVaR: measures expected loss *given* a minimum loss equal to the VaR
- Equals **expected value** of the *tail* of the loss distribution:
 - $\text{CVaR}(\alpha) := \frac{1}{1 - \alpha} \mathbb{E} \int_{\text{VaR}(\alpha)}^{\bar{x}} x f(x) dx,$
 - $f(\cdot)$ = loss distribution pdf
 - \bar{x} = upper bound of the loss (can be infinity)
 - $\text{VaR}(\alpha)$ = VaR at the α confidence level.

- **Forex Example:**
 - 95% CVaR = expected loss for 5% of cases when portfolio value *smaller* than EUR 40



Deriving the VaR

1. Specify confidence level, e.g. 95% (0.95)
2. Create Series of `loss` observations
3. Compute `loss.quantile()` at specified confidence level
4. $\text{VaR} = \text{computed .quantile()}$ at desired confidence level
5. `scipy.stats` loss distribution: percent point function `.ppf()` can also be used

```
loss = pd.Series(observations)
VaR_95 = loss.quantile(0.95)
print("VaR_95 = ", VaR_95)
```

```
VaR_95 = 1.6192834157254088
```

Deriving the CVaR

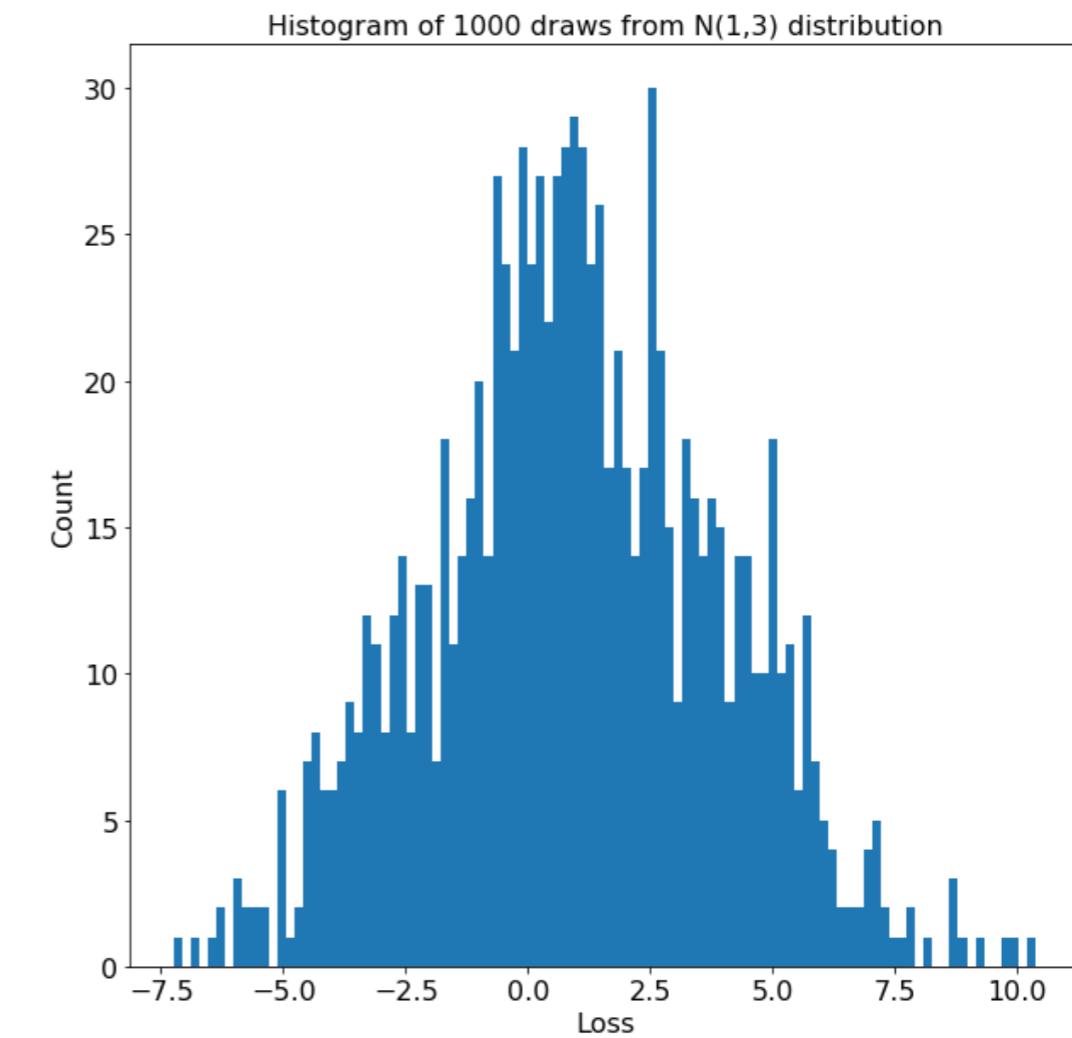
1. Specify confidence level, e.g. 95% (0.95)
2. Create or use sample from loss distribution
3. Compute VaR at a specified confidence level, e.g. 0.95.
4. Compute CVaR as expected loss (Normal distribution: `scipy.stats.norm.expect()` does this).

```
losses = pd.Series(scipy.stats.norm.rvs(size=1000))
VaR_95 = scipy.stats.norm.ppf(0.95)
CVaR_95 = (1/(1 - 0.95))*scipy.stats.norm.expect(lambda x: x, lb = VaR_95)
print("CVaR_95 = ", CVaR_95)
```

```
CVaR_95 = 2.153595332530393
```

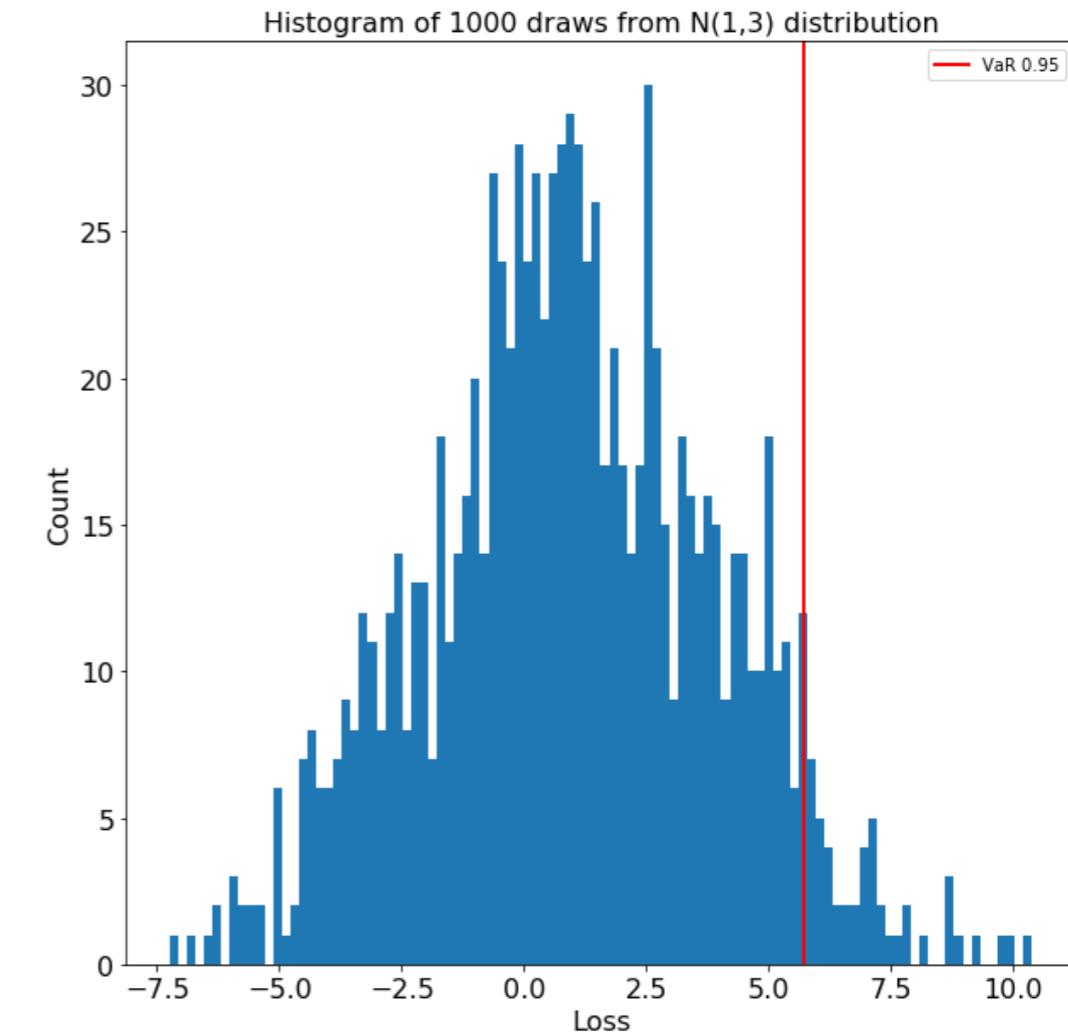
Visualizing the VaR

- Loss distribution histogram for 1000 draws from $N(1,3)$



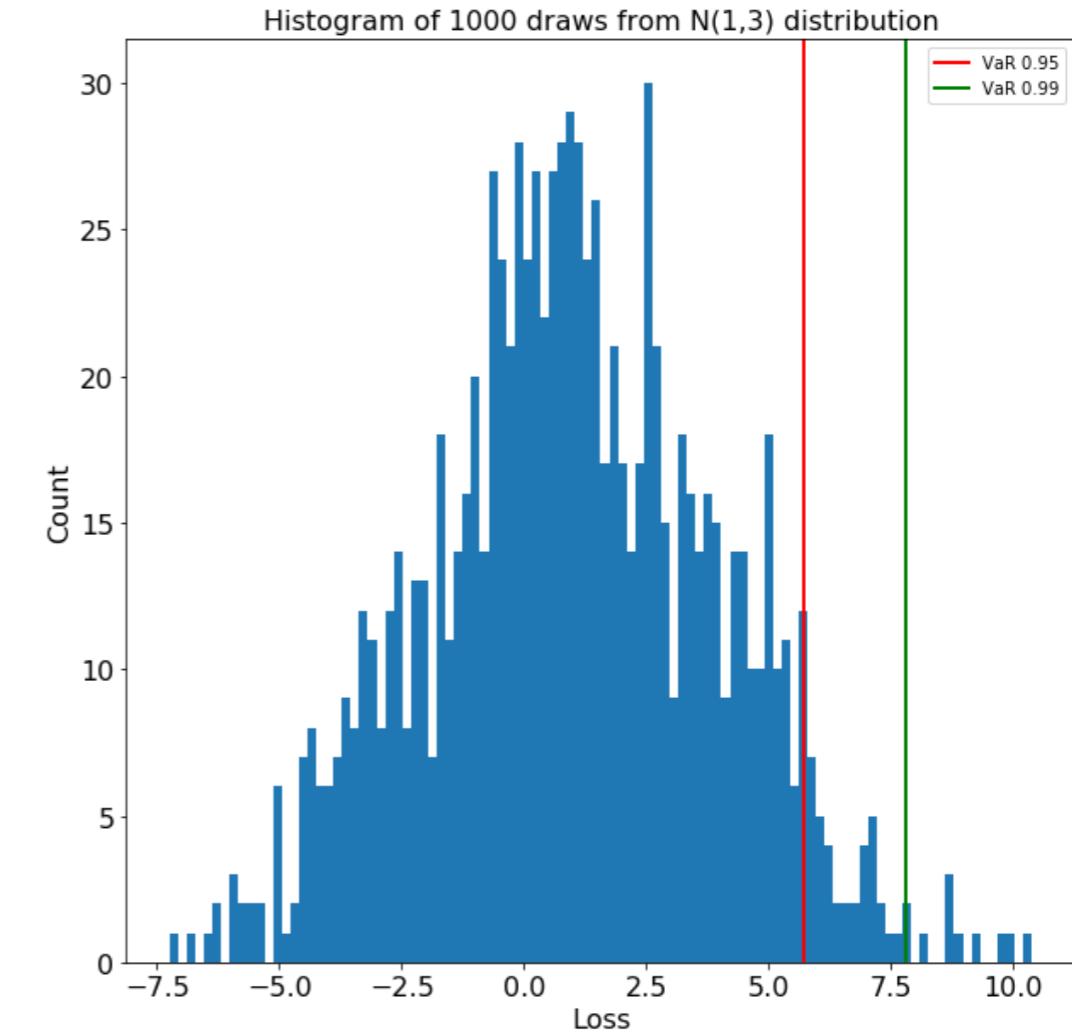
Visualizing the VaR

- Loss distribution histogram for 1000 draws from $N(1,3)$
 - $VaR_{95} = 5.72$, i.e. VaR at 95% confidence



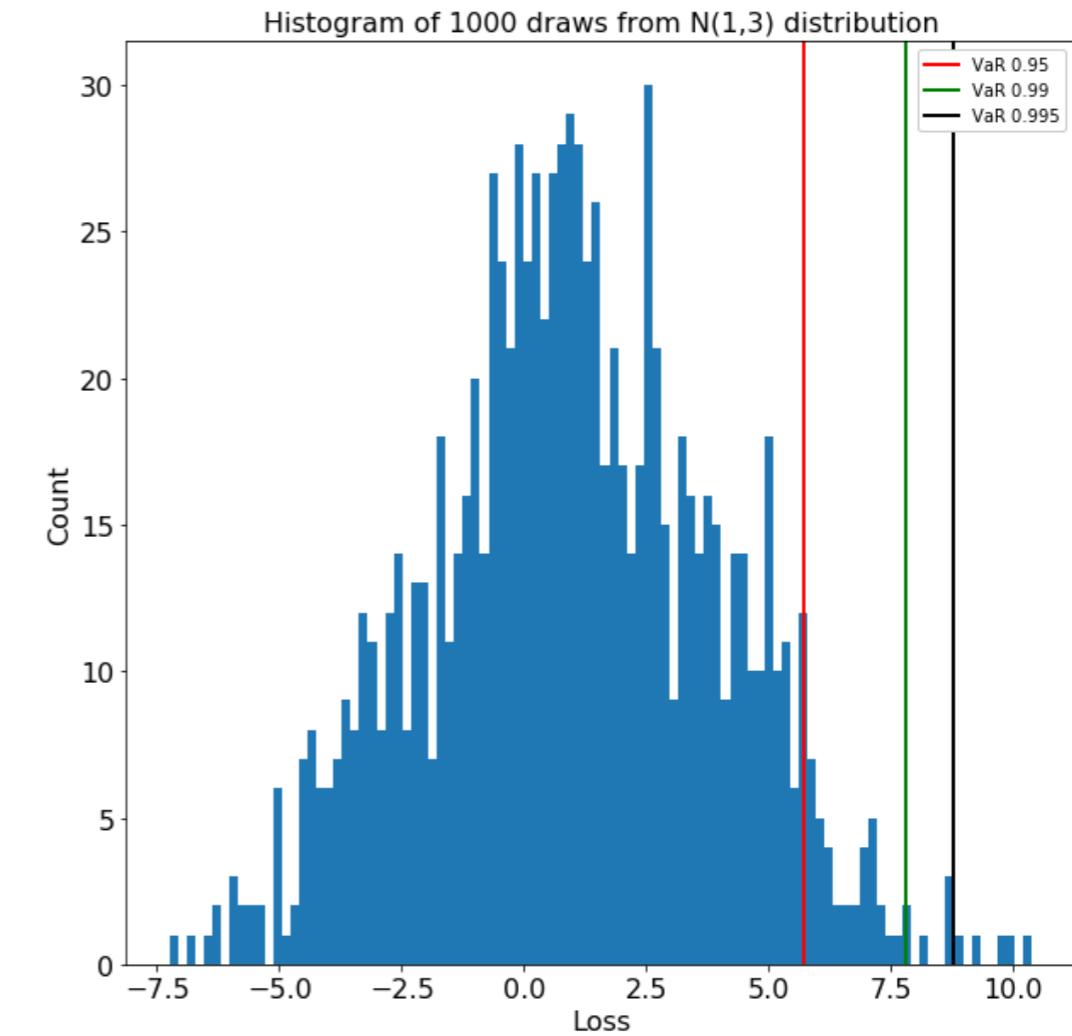
Visualizing the VaR

- Loss distribution histogram for 1000 draws from $N(1,3)$
 - $VaR_{95} = 5.72$, i.e. VaR at 95% confidence
 - $VaR_{99} = 7.81$, i.e. VaR at 99% confidence



Visualizing the VaR

- Loss distribution histogram for 1000 draws from $N(1,3)$
 - $VaR_{95} = 5.72$, i.e. VaR at 95% confidence
 - $VaR_{99} = 7.81$, i.e. VaR at 99% confidence
 - $VaR_{99.5} = 8.78$, i.e. VaR at 99.5% confidence
- The VaR measure *increases* as the confidence level *rises*

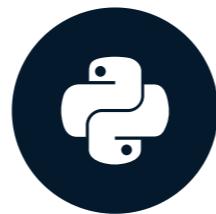


Let's practice!

QUANTITATIVE RISK MANAGEMENT IN PYTHON

Risk exposure and loss

QUANTITATIVE RISK MANAGEMENT IN PYTHON



Jamsheed Shorish
Computational Economist

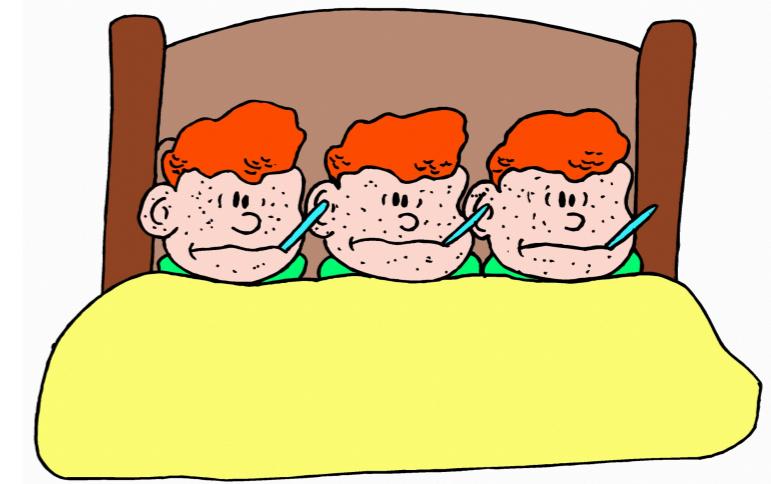
A vacation analogy

- Hotel reservations for vacation
- Pay in advance, before stay
 - Low room rate
 - **Non-refundable:** cancellation fee = 100% of room rate
- Pay after arrival
 - High room rate
 - **Partially refundable:** cancellation fee of 20% of room rate



Deciding between options

- What determines your decision?
 1. Chance of negative shock: illness, travel disruption, weather
 - **Probability of loss**
 2. Loss associated with shock: amount or conditional amount
 - e.g. **VaR, CVaR**
 3. Desire to avoid shock: personal feeling
 - **Risk tolerance**



Risk exposure and VaR

- **Risk exposure:** probability of loss x loss measure
 - Loss measure: e.g. VaR
- **10% chance of canceling vacation:** $P(\text{Illness}) = 0.10$
- **Non-refundable:**
 - Total non-refundable hotel cost: € 500
 - VaR at 90% confidence level: € 500
- **Partially refundable:**
 - Refundable hotel cost: € 550
 - VaR at 90% confidence level: 20% cancellation fee x € 550 = € 110

Calculating risk exposure

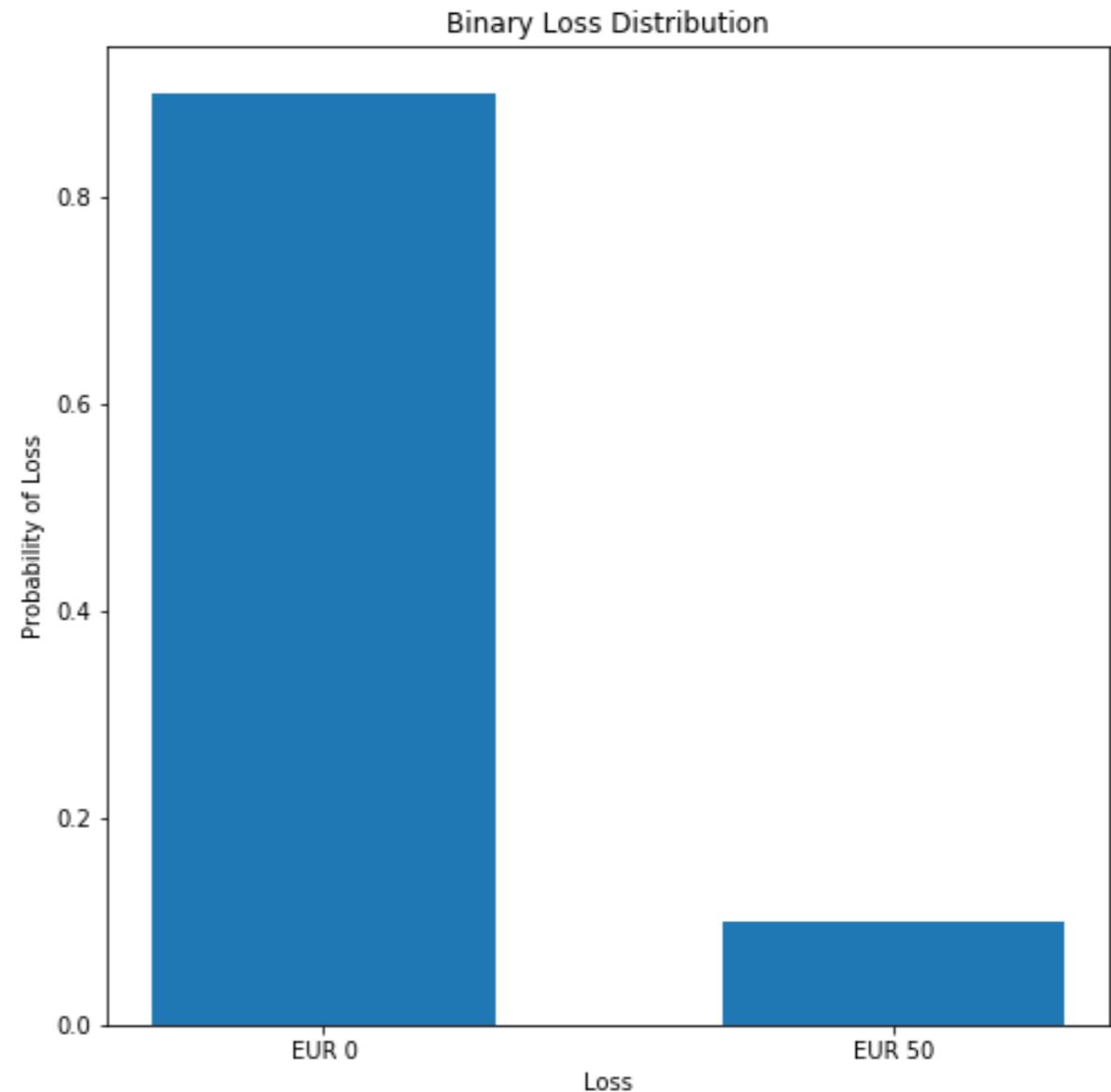
- **Non-refundable exposure ("nr"):**
 - $P(\text{illness}) \times \text{VaR}_{0.90}^{\text{nr}} = 0.10 \times € 500 = € 50.$
- **Partially refundable exposure ("pr"):**
 - $P(\text{illness}) \times \text{VaR}_{0.90}^{\text{pr}} = 0.10 \times € 110 = € 11.$
- **Difference in risk exposure:** $€ 50 - € 11 = € 39.$
- **Total price difference between offers:** $€ 550 - € 500 = € 50.$
- *Risk tolerance:* is paying $€ 50$ *more* worth **avoiding** $€ 39$ of additional exposure?

Risk tolerance and risk appetite

- **Risk-neutral:** only expected values matter
 - $\text{€ 39} < \text{€ 50} \Rightarrow$ prefer **non-refundable** option
- **Risk-averse:** uncertainty itself carries a cost
 - $\text{€ 39} < \text{€ 50} \Rightarrow$ prefer **partially refundable** option
- Enterprise/institutional risk management: preferences as ***risk appetite***
- Individual investors: preferences as ***risk tolerance***

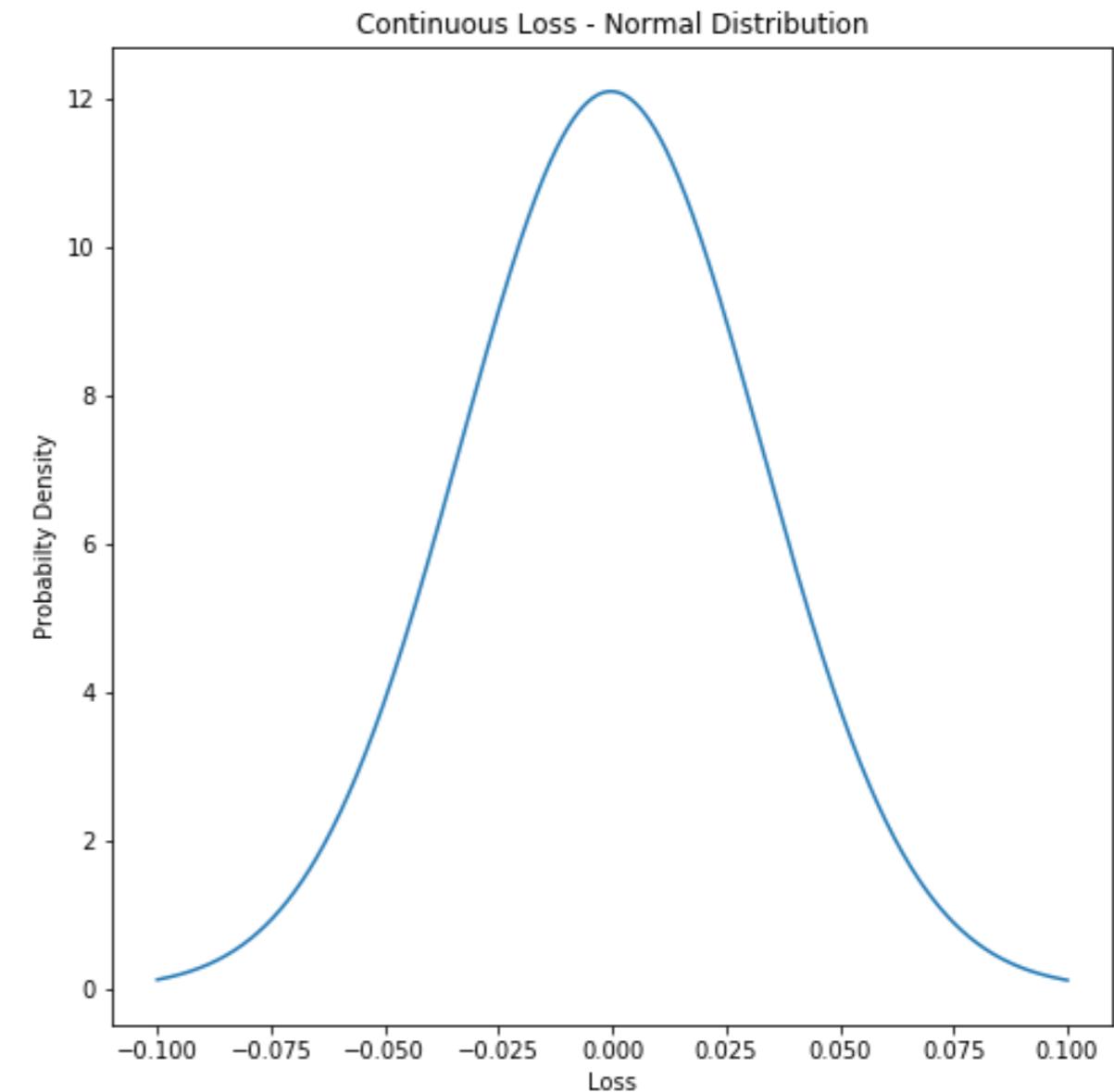
Loss distribution - discrete

- Risk exposure *depends upon* loss distribution (probability of loss)
- **Vacation example:** 2 outcomes from random risk factor



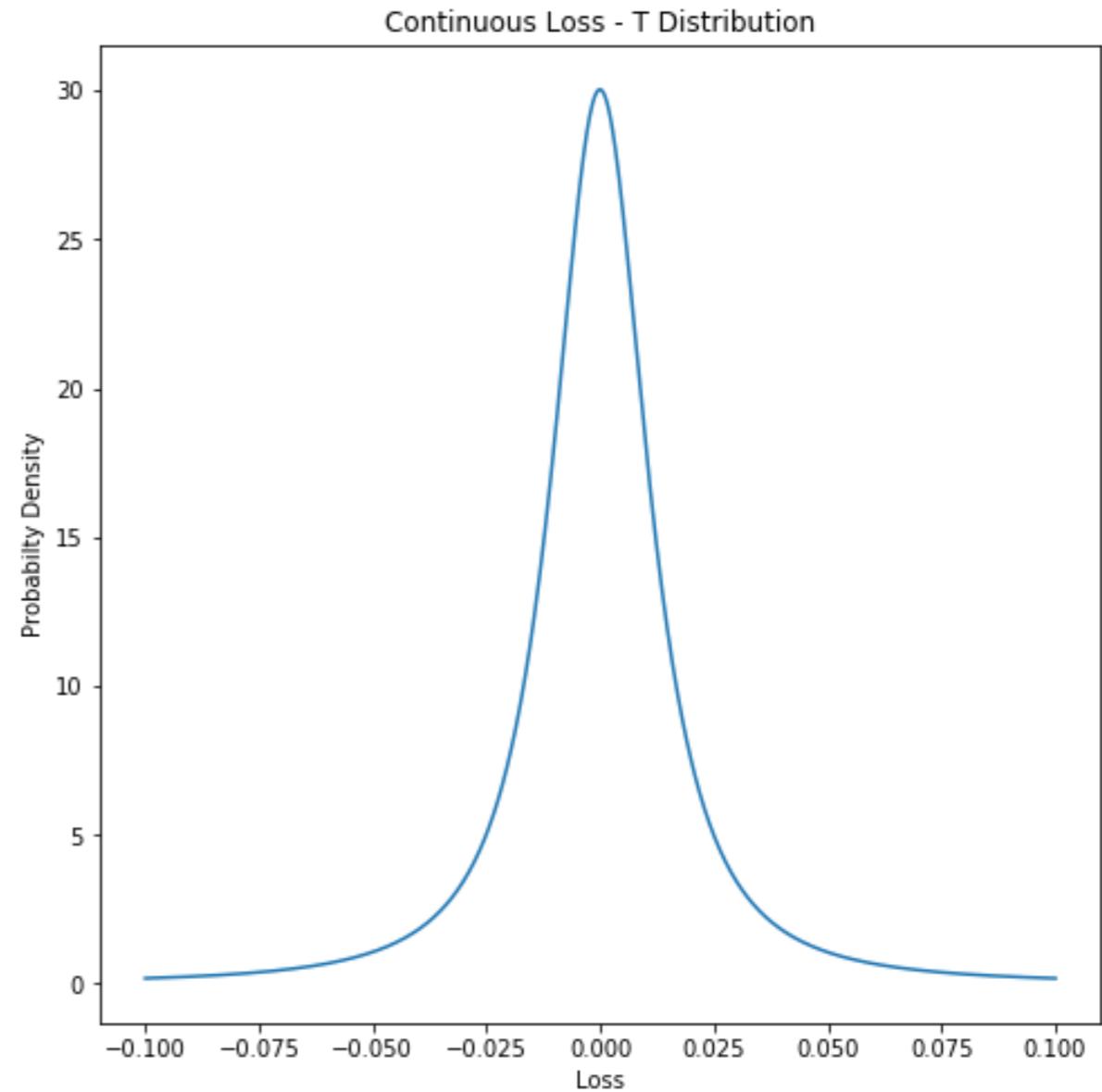
Loss distribution - continuous

- Risk exposure *depends upon* loss distribution (probability of loss)
- **Vacation example:** 2 outcomes from random risk factor
- More generally: *continuous* loss distribution
 - **Normal distribution:** good for large samples



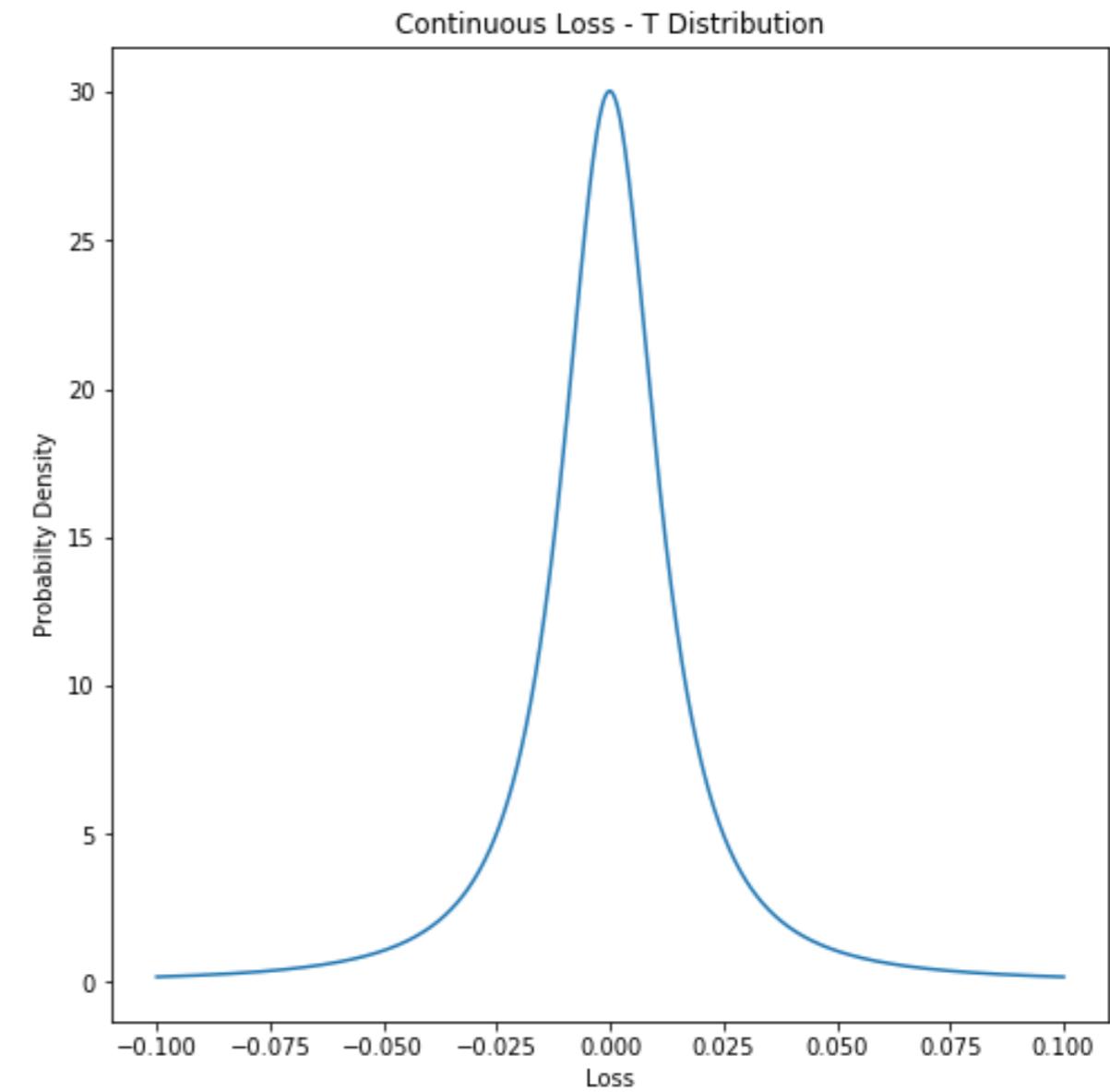
Loss distribution - continuous

- Risk exposure *depends upon* loss distribution (probability of loss)
- **Vacation example:** 2 outcomes from random risk factor
- More generally: *continuous* loss distribution
 - **Normal distribution:** good for large samples
 - **Student's t-distribution:** good for smaller samples



Primer: Student's t-distribution

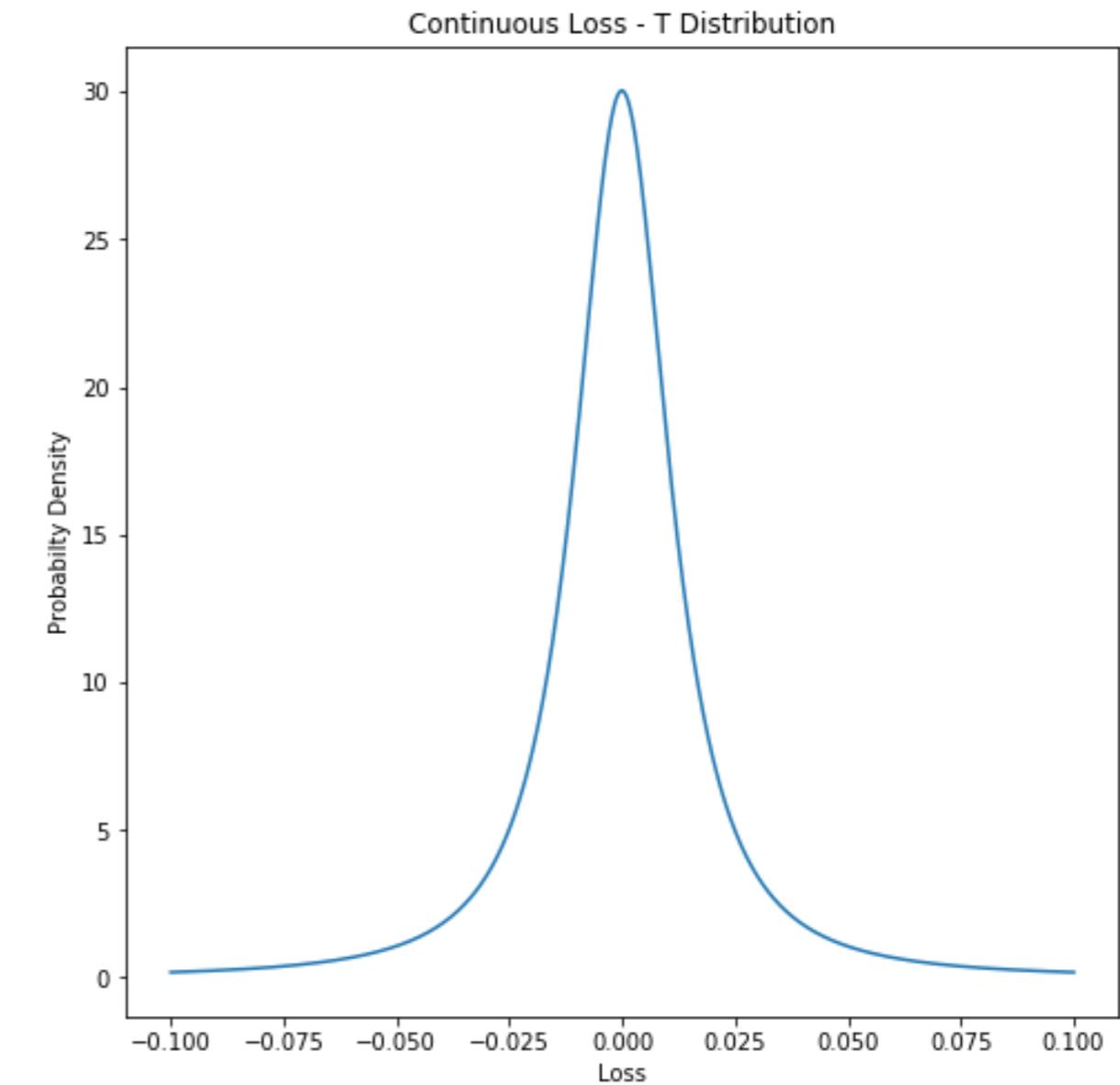
- Also referred to as **T** distribution
- Has "fatter" tails than Normal for small samples
 - Similar to portfolio returns/losses
- As sample size grows, **T** converges to Normal distribution



T distribution in Python

- Example: compute 95% VaR from T distribution
 - Import t distribution from scipy.stats
 - Fit portfolio_loss data using t.fit()

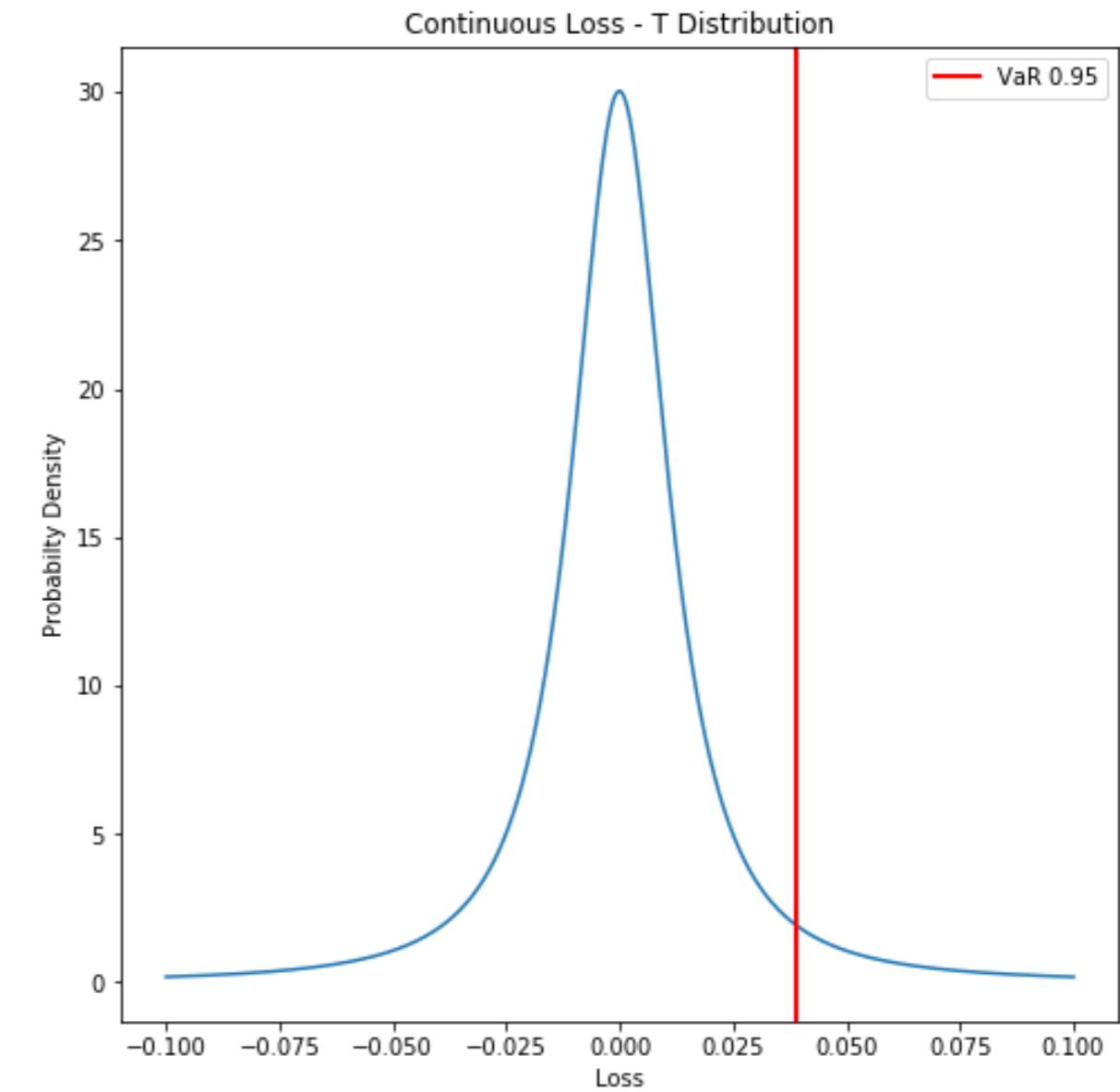
```
from scipy.stats import t  
params = t.fit(portfolio_losses)
```



T distribution in Python

- Example: compute 95% VaR from T distribution
 - Import `t` distribution from `scipy.stats`
 - Fit `portfolio_loss` data using `t.fit()`
 - Compute percent point function with `.ppf()` to find VaR

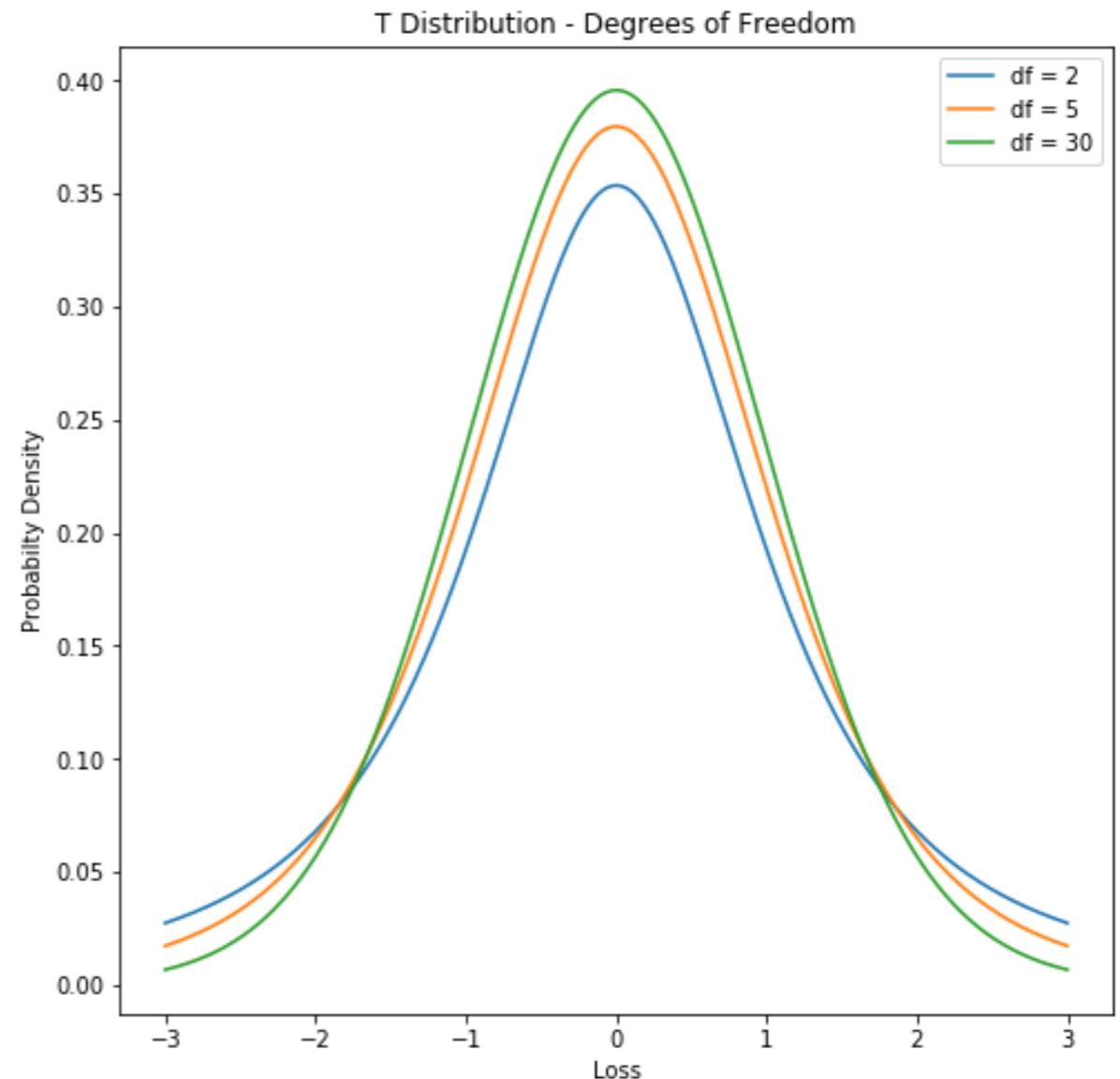
```
from scipy.stats import t
params = t.fit(portfolio_losses)
VaR_95 = t.ppf(0.95, *params)
```



Degrees of freedom

- **Degrees of freedom (df)**: number of independent observations
- *Small df*: "fat tailed" T distribution
- *Large df*: Normal distribution

```
x = np.linspace(-3, 3, 100)
plt.plot(x, t.pdf(x, df = 2))
plt.plot(x, t.pdf(x, df = 5))
plt.plot(x, t.pdf(x, df = 30))
```

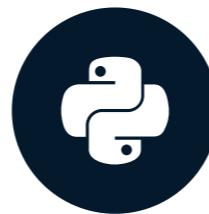


Let's practice!

QUANTITATIVE RISK MANAGEMENT IN PYTHON

Risk management using VaR & CVaR

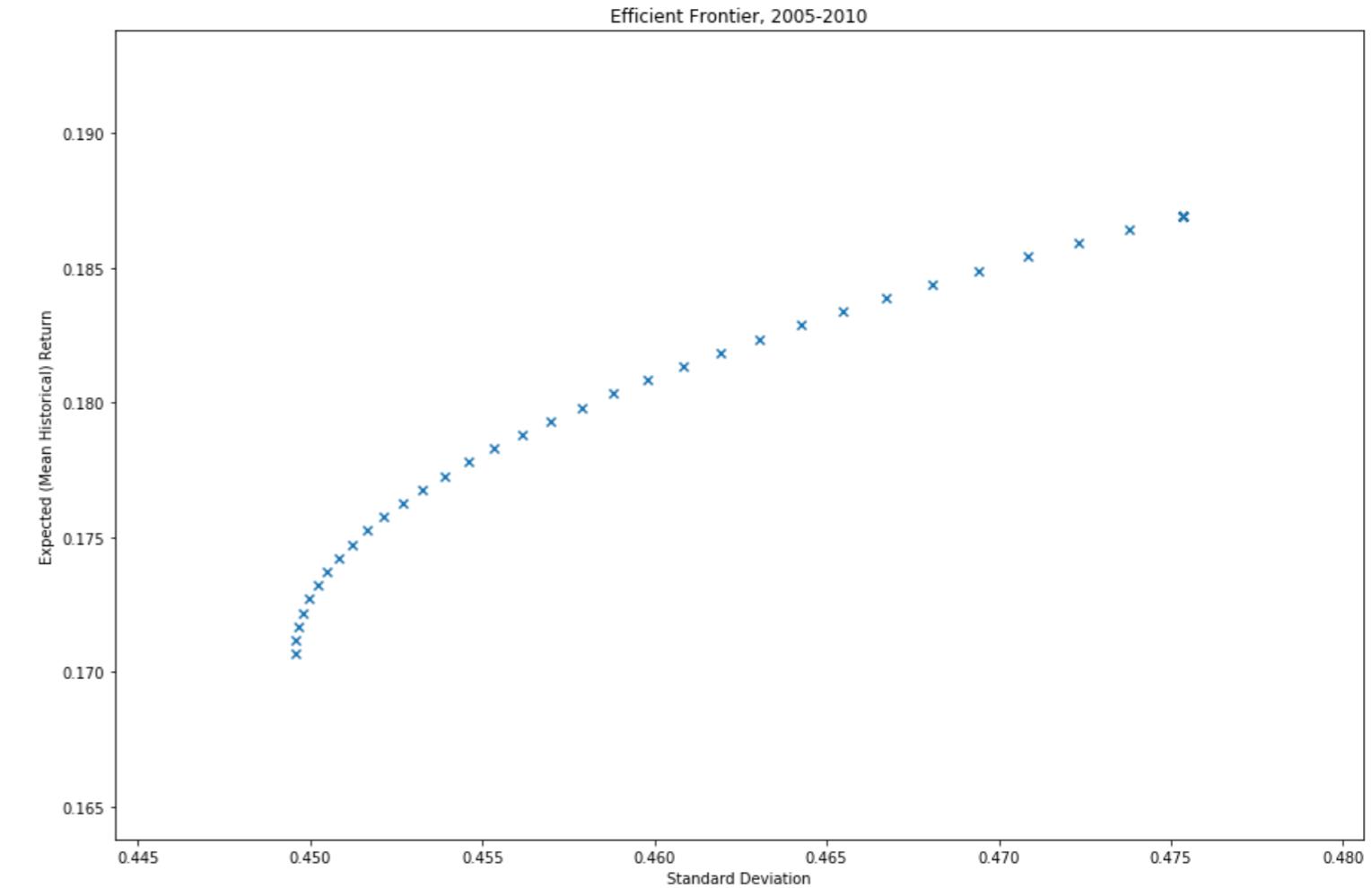
QUANTITATIVE RISK MANAGEMENT IN PYTHON



Jamsheed Shorish
Computational Economist

Risk management via modern portfolio theory

- **Efficient Portfolio**
 - Portfolio weights maximize return given risk level
- **Efficient Frontier:** locus of (risk, return) points generated by different efficient portfolios
 - Each point = portfolio weight optimization
- Creation of efficient portfolio/frontier:
Modern Portfolio Theory



Incorporating Value at Risk into MPT

- **Modern Portfolio Theory (MPT)**: "mean-variance" optimization
 - Highest expected return
 - Risk level (volatility) is given
 - *Objective function*: expected return
- **VaR/CVaR**: measure risk over distribution of *loss*
- Adapt MPT to optimize over *loss distribution* vs. expected return

A new objective: minimize CVaR

- Change objective of portfolio optimization
 - **mean-variance objective**: maximize expected mean return
 - **CVaR objective**: minimize expected conditional loss at a given confidence level
- **Example:** Loss distribution
 - **VaR**: maximum loss with 95% confidence
 - **CVaR**: expected loss given *at least* VaR loss (worst 5% of cases)
 - Optimization: portfolio weights **minimizing** CVaR
- Find lowest expected loss in worst $100\% - 95\% = 5\%$ of possible outcomes

The risk management problem

- Select optimal portfolio weights w^* as solution to

$$\min_w \frac{1}{1 - \alpha} \mathbb{E} \int_{\text{VaR}(\alpha)}^{\infty} x(w) f(x(w)) dx$$

with

$$\sum_i w_i = 1$$

- Recall: $f(x)$ = probability density function of portfolio **loss**
- PyPortfolioOpt: select minimization of CVaR as new objective

CVaR minimization using PyPortfolioOpt

- Create an `EfficientFrontier` object with an efficient covariance matrix `e_cov`
- Import built-in objective function that minimizes CVaR, `.negative_cvar()` from `pypfopt.objective_functions` module
- Compute optimal portfolio weights using `.custom_objective()` method
 - Arguments of `.negative_cvar()` added to `.custom_objective()`.

```
ef = pypfopt.efficient_frontier.EfficientFrontier(None, e_cov)
from pypfopt.objective_functions import negative_cvar
optimal_weights = ef.custom_objective(negative_cvar, returns)
```

Negative CVaR?

- Seek minimum CVaR portfolio at given significance level $1 - \alpha$.
- **Same as** finding portfolio that *maximizes returns* in worst $1 - \alpha$ cases
- **Question:** expect objective function to return *positive* number, or *negative*?

Negative CVaR?

- Optimization can either:
 - *Maximize* something, or
 - *Minimize the negative* of something
- PyPortfolioOpt solver: **minimizes** by default
 - So objective function needs to be *negative* of CVaR **returns**
 - Gives *same answer* as minimizing CVaR **losses**
- Term "negative CVaR" is a misnomer: CVaR is an expected **loss**

Mean-variance vs. CVaR risk management

- Mean-variance minimum volatility portfolio, 2005-2010 investment bank assets

```
ef = EfficientFrontier(None, e_cov)
min_vol_weights = ef.min_volatility()
print(min_vol_weights)
```

```
{'Citibank': 0.0,
'Morgan Stanley': 5.0784330940519306e-18,
'Goldman Sachs': 0.6280157234640608,
'J.P. Morgan': 0.3719842765359393}
```

Mean-variance vs. CVaR risk management

- CVaR-minimizing portfolio, 2005-2010 investment bank assets

```
min_cvar_weights = ef.custom_objective(negative_cvar, returns)  
print(min_cvar_weights)
```

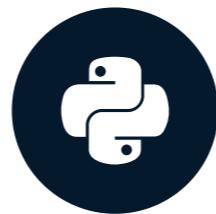
```
{'Citibank': 0.25204510921196405,  
'Morgan Stanley': 0.24871969691415985,  
'Goldman Sachs': 0.2485991950454842,  
'J.P. Morgan': 0.25076949897532647}
```

Let's practice!

QUANTITATIVE RISK MANAGEMENT IN PYTHON

Portfolio hedging: offsetting risk

QUANTITATIVE RISK MANAGEMENT IN PYTHON



Jamsheed Shorish
Computational Economist

Portfolio stability

- **VaR/CVaR:** potential portfolio loss for given confidence level
- **Portfolio optimization:** 'best' portfolio weights
 - *But volatility is still present!*
- **Institutional investors:** *stability* of portfolio against volatile changes
 - **Pension funds:** c. USD 20 trillion

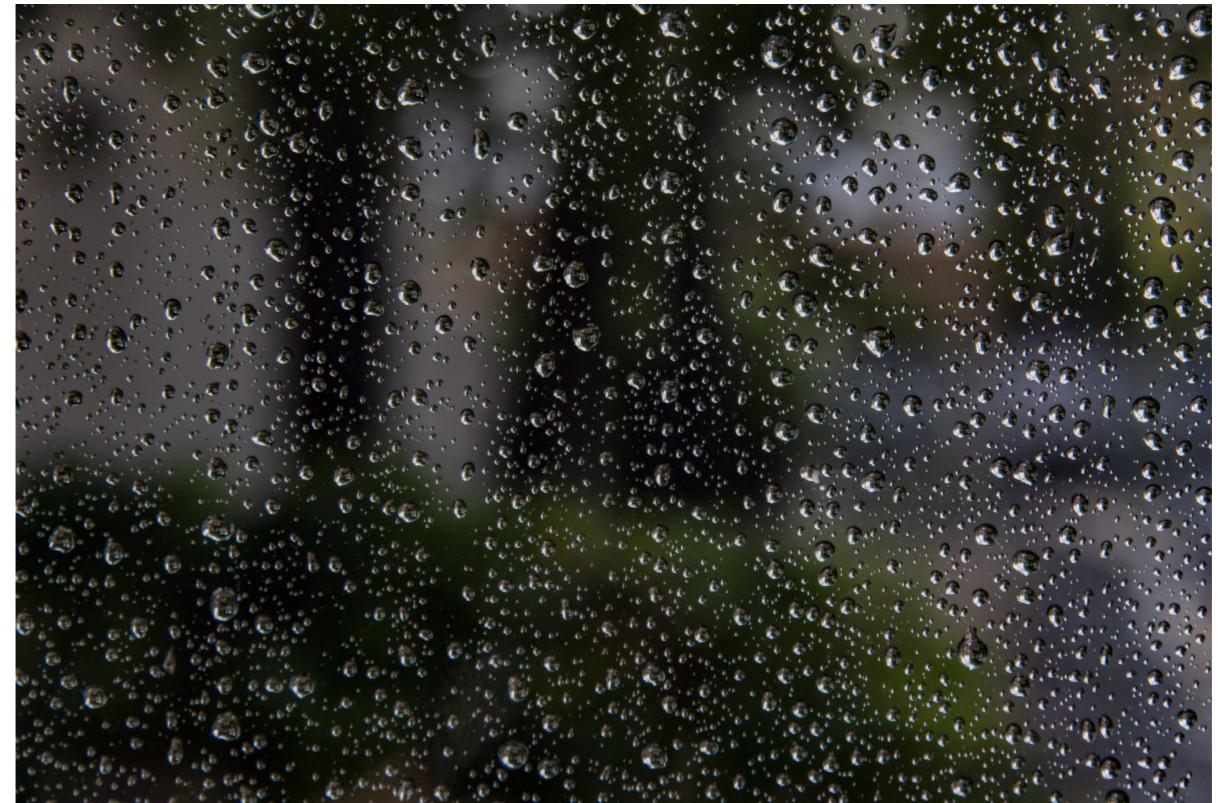
Rainy days, sunny days

- **Investment portfolio:** sunglasses company
 - **Risk factor:** weather



Rainy days, sunny days

- **Investment portfolio:** sunglasses company
 - **Risk factor:** weather (**rain**)
 - More rain => lower company value
 - Lower company value => lower stock price
 - Lower stock price => *lower portfolio value*



Rainy days, sunny days

- **Investment portfolio:** sunglasses company
 - **Risk factor:** weather (**rain**)
 - More rain => lower company value
 - Lower company value => lower stock price
 - Lower stock price => *lower portfolio value*
- **Second opportunity:** umbrella company
 - More rain => **more value!**



Rainy days, sunny days

- **Investment portfolio:** sunglasses company
 - **Risk factor:** weather (**rain**)
 - More rain => lower company value
 - Lower company value => lower stock price
 - Lower stock price => *lower portfolio value*
- Second opportunity: umbrella company
 - More rain => **more** value!
- Portfolio: sunglasses & umbrellas, *more stable*
 - Volatility of rain is *offset*



Hedging

- **Hedging:** offset volatility with another asset
- Crucial for institutional investor risk management
- Additional return stream moving *opposite* to portfolio
- Used in pension funds, ForEx, futures, derivatives...
 - 2019: hedge fund market c. USD **3.6 trillion**

Hedge instruments: options

- **Derivative:** hedge instrument
- **European option:** very popular derivative
 - European **call** option: right (not obligation) to purchase stock at fixed price X on date M
 - European **put** option: right (not obligation) to sell stock at fixed price X on date M
 - Stock = "underlying" of the option
 - Current market price S = *spot price*
 - X = *strike price*
 - M = *maturity date*

Black-Scholes option pricing

- Option value changes when price of underlying changes => can be used to **hedge risk**
- Need to **value** option: requires assumptions about market, underlying, interest rate, etc.
- **Black-Scholes** option pricing formula: Fisher Black & Nobel Laureate Myron Scholes (1973)
 - Requires for each time t :
 - spot price S
 - strike price X
 - time to maturity $T := M - t$
 - risk-free interest rate r
 - volatility of underlying returns σ (standard deviation)

¹ Black, F. and M. Scholes (1973). "The Pricing of Options and Corporate Liabilities", Journal of Political Economy vol 81 no. 3, pp. 637–654.{{3}}

Black-Scholes formula assumptions

- Market structure
 - Efficient markets
 - No transactions costs
 - Risk-free interest rate
- Underlying stock
 - No dividends
 - Normally distributed returns
- Online calculator: <http://www.math.drexel.edu/~pg/fin/VanillaCalculator.html>
- Python function `black_scholes()` : source code link available in the exercises

Computing the Black-Scholes option value

- Black-Scholes option pricing formula `black_scholes()`
- Required parameters: S , X , T (in fractions of a year), r , σ
- Use the desired `option_type` ('call' or 'put')

```
S = 70; X = 80; T = 0.5; r = 0.02; sigma = 0.2
option_value = black_scholes(S, X, T, r, sigma, option_type = "put")
print(option_value)
```

```
10.31222171237868
```

Hedging a stock position with an option

- Hedge stock with European put option: underlying is same as stock in portfolio
- Spot price S falls ($\Delta S < 0$) => option value V rises ($\Delta V > 0$)
- *Delta* of an option: $\Delta := \frac{\partial V}{\partial S}$
- Hedge one share with $\frac{1}{\Delta}$ options
- **Delta neutral:** $\Delta S + \frac{\Delta V}{\Delta} = 0$; stock is hedged!
- Python function `bs_delta()` : computes the option delta
 - Link to source available in the exercises

Let's practice!

QUANTITATIVE RISK MANAGEMENT IN PYTHON