Applied Quantitative Investment Management

Lecture 1: Introduction and Python setup

Anton Vorobets

Agenda

- How the book and this course is different (Preface)
- Introduction to the investment framework and its core methods (Chapter 1.1)
- Preparing for the next lecture(s).
- Setting up a proper Python development environment, so you can explore the code.

My experience

Professional:

2021-present: Founder & CEO @ **Fortitudo Technologies**

2016-2021: Portfolio Construction and Asset Allocation @ Danske Bank Asset Management

2015-2016: Equity Derivatives Strategy @ Nordea Markets

2011-2015: Teaching Assistant in Mathematics, Statistics and Econometrics @ Aarhus University and University of Southern Denmark

Education:

MSc Quantitative Finance and Machine Learning,

Aarhus University and University of Copenhagen

Substack:

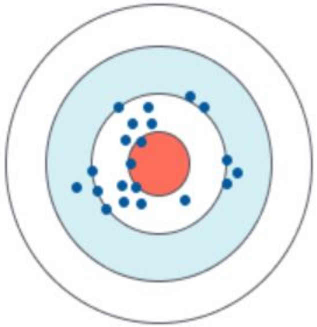
LinkedIn:

Scientific quality of finance and economics academia

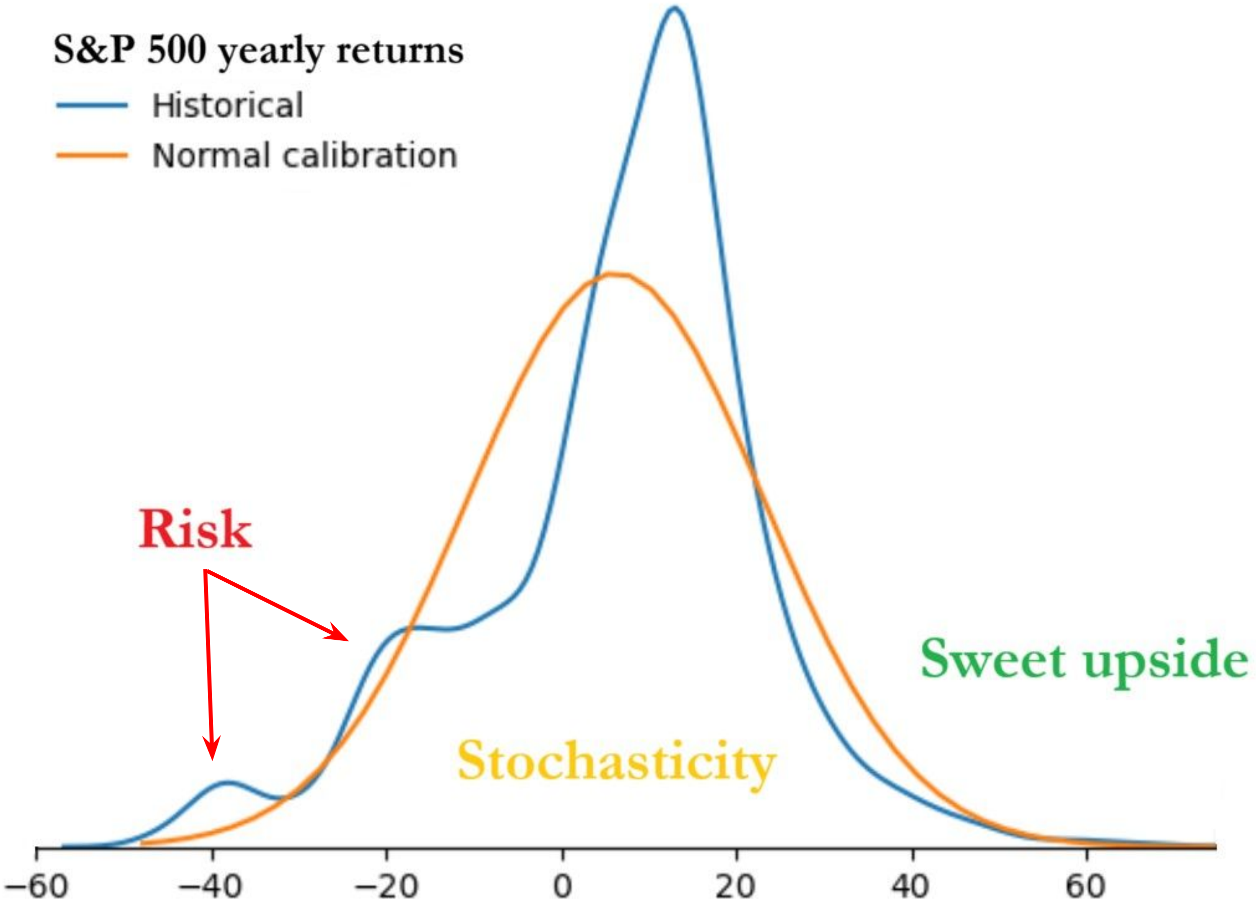
Finance and economics academia is exactly wrong

Exactly wrong

Approximately correct



Problem with mean-variance, Black-Litterman, etc.



Market representation

$$R = \begin{pmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,I} \\ R_{2,1} & R_{2,2} & \cdots & R_{2,I} \\ \vdots & \vdots & \ddots & \vdots \\ R_{S,1} & R_{S,2} & \cdots & R_{S,I} \end{pmatrix} \quad p = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_S \end{pmatrix}$$

$$p_s \in (0,1] \quad \sum_{s=1}^{S} p_s = 1$$

VS

$$\mu \in \mathbb{R}^I \quad \Sigma \in \mathbb{R}^{I \times I}$$

Market representation

	DM Gov	Corp IG	Corp HY	EM Gov	DM Equities	EM Equities	Private Equity	Infrastructure	Real Estate	Hedge Funds	Put option	p	q
0	-2.901372	1.515755	7.243079	9.858614	27.987974	-7.091294	12.952589	0.607117	-2.051072	10.892261	-1.875000	0.01	0.009528
1	-4.033522	-5.507238	-17.472262	-18.084804	-20.901868	-42.943386	-24.630603	-4.877102	-6.808599	-8.255685	9.026868	0.01	0.010300
2	-2.112649	3.396251	3.433316	-2.171095	-0.207091	-24.640181	-4.598131	10.126871	-2.997219	2.751281	-1.875000	0.01	0.007035
3	4.953209	3.406907	27.820853	4.436701	25.658191	46.631981	75.017185	1.123291	10.170868	26.980165	-1.875000	0.01	0.002785
4	4.077621	-2.803989	1.067991	1.161560	19.527465	10.567691	27.660339	-4.110836	3.671376	10.251468	-1.875000	0.01	0.017757
5	5.028862	7.579274	14.766076	6.974784	-0.907956	-21.555335	5.031809	3.706813	-1.695980	0.211889	-1.875000	0.01	0.008335
6	-2.700348	-0.810961	-9.501812	-6.684766	1.070018	-6.555039	2.339627	6.119593	3.027697	5.498058	-1.875000	0.01	0.007617
7	3.976312	5.930553	4.328590	1.981325	13.535295	21.631855	-0.783067	11.487004	0.858899	10.237277	-1.875000	0.01	0.014521
8	-4.416313	-4.715980	-8.971976	-6.600808	-0.293693	-4.104135	17.010508	-14.610564	3.032070	3.442459	-1.875000	0.01	0.010635
9	2.115861	6.387309	14.575082	15.427097	11.223288	29.653759	29.527585	10.523831	5.674087	9.411179	-1.875000	0.01	0.018593
10	0.035257	1.025236	-5.285266	0.624084	-10.851014	-30.276292	-20.482211	-2.739756	0.922560	-6.129087	-1.023986	0.01	0.004728
11	1.974382	1.414184	16.825757	-2.814275	11.233995	19.782523	15.896183	-2.049651	0.240350	13.847232	-1.875000	0.01	0.012243
12	5.280698	0.871584	11.513783	3.309399	25.206057	18.720860	81.037608	9.508263	24.317649	19.498595	-1.875000	0.01	0.009221
13	2.257190	0.978109	4.755107	3.687936	1.943316	-2.218547	-11.355873	8.983795	6.558238	2.958365	-1.875000	0.01	0.013361

Views and stress-testing with Entropy Pooling

$$q = \underset{x}{\operatorname{argmin}} \left\{ x^T (\ln x - \ln p) \right\}$$

$$\text{s.t.} \quad Gx \leq h \quad Ax = b$$

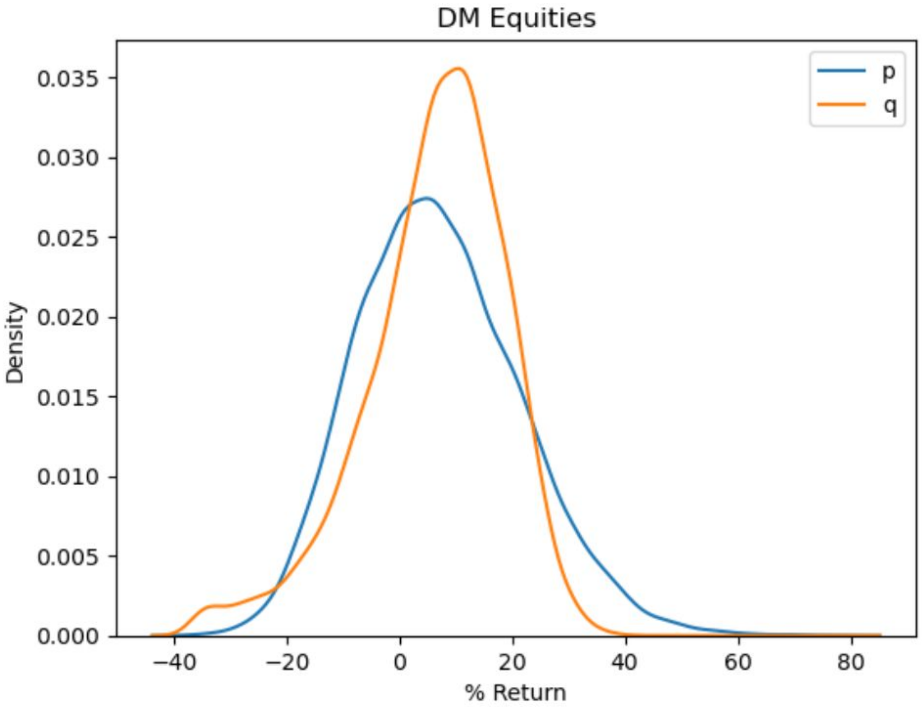
$$x > 0 \quad \sum_{s=1}^{S} x_s = 1$$

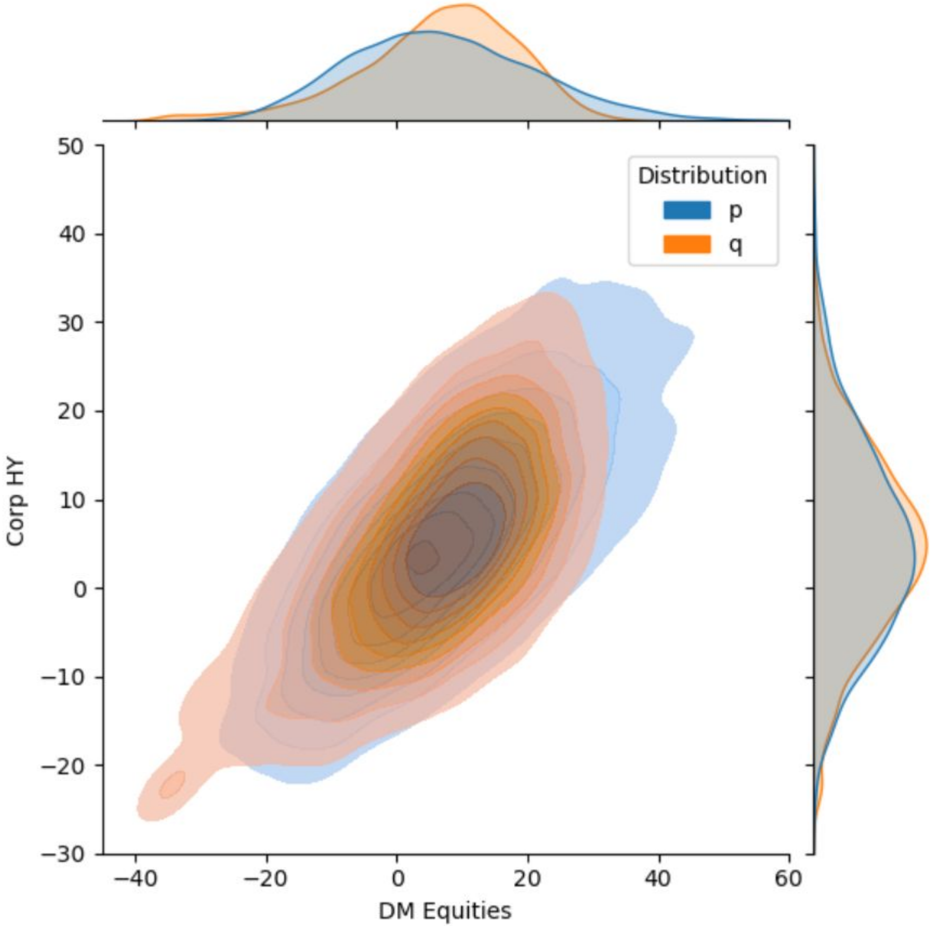
VS

$$\mu_{BL} = \left[(\tau\Sigma)^{-1} + P^T \Omega^{-1} P \right]^{-1} \left[(\tau\Sigma)^{-1} \Pi + P^T \Omega^{-1} Q \right]$$

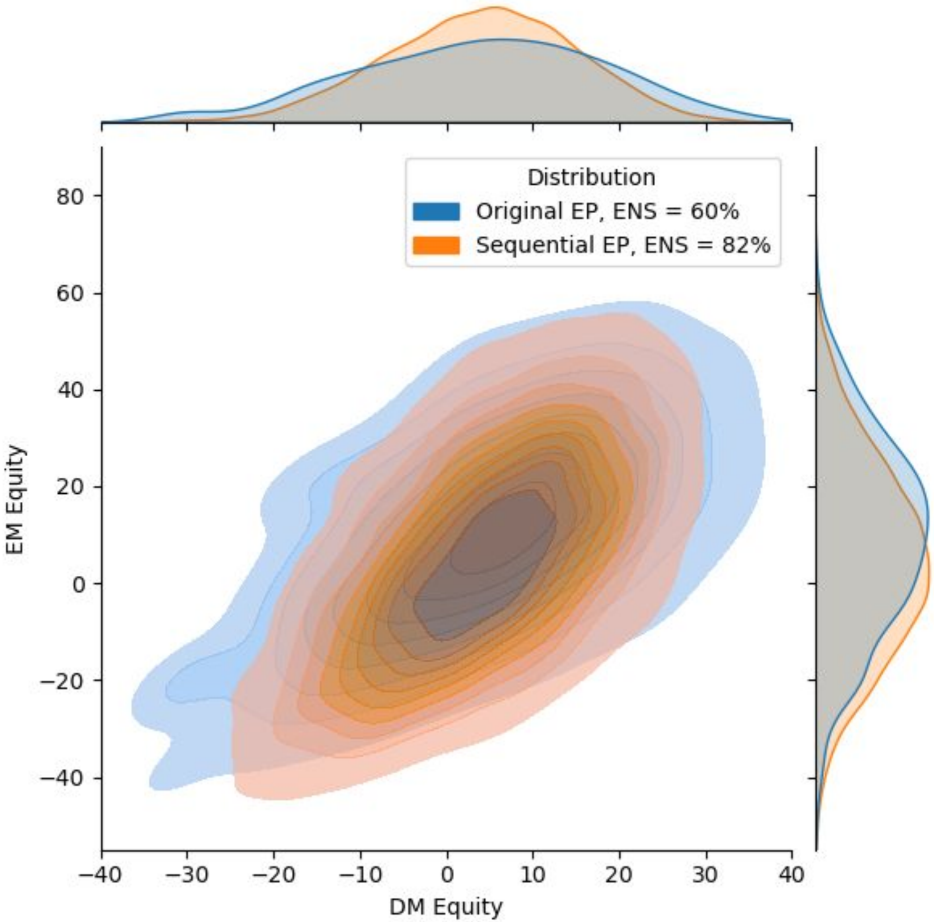
$$\Sigma_{BL} = \Sigma + \left[(\tau\Sigma)^{-1} + P^T \Omega^{-1} P \right]$$

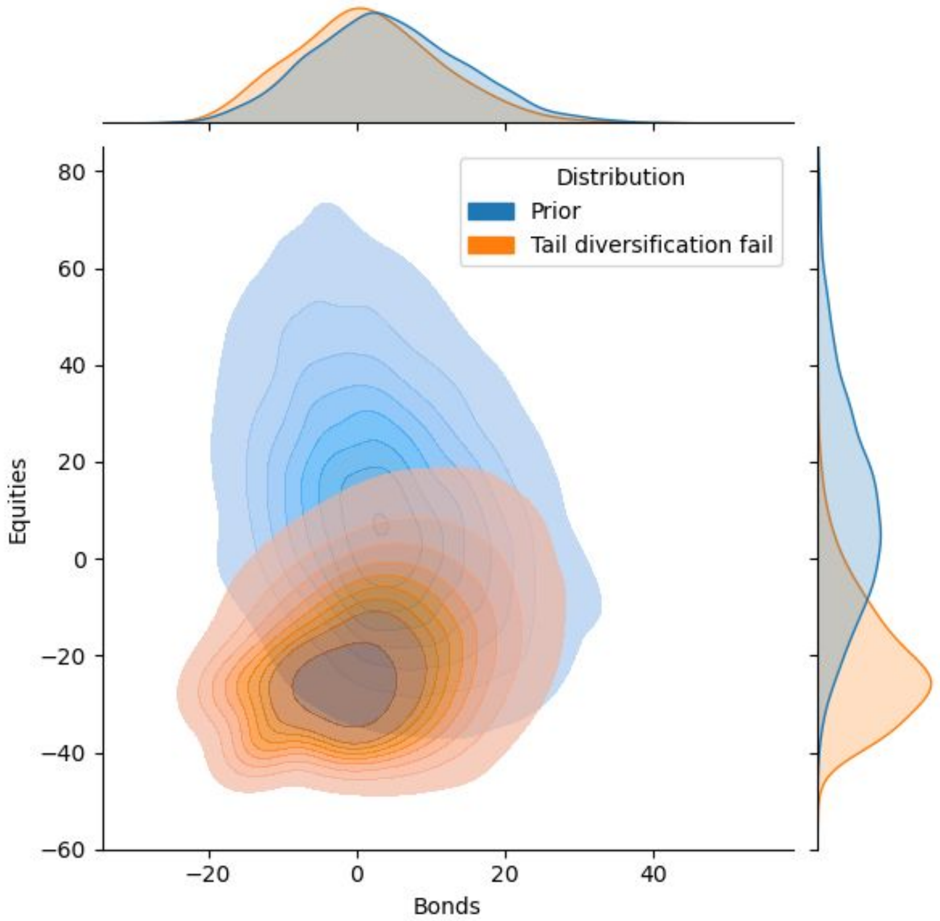
Quick Entropy Pooling example



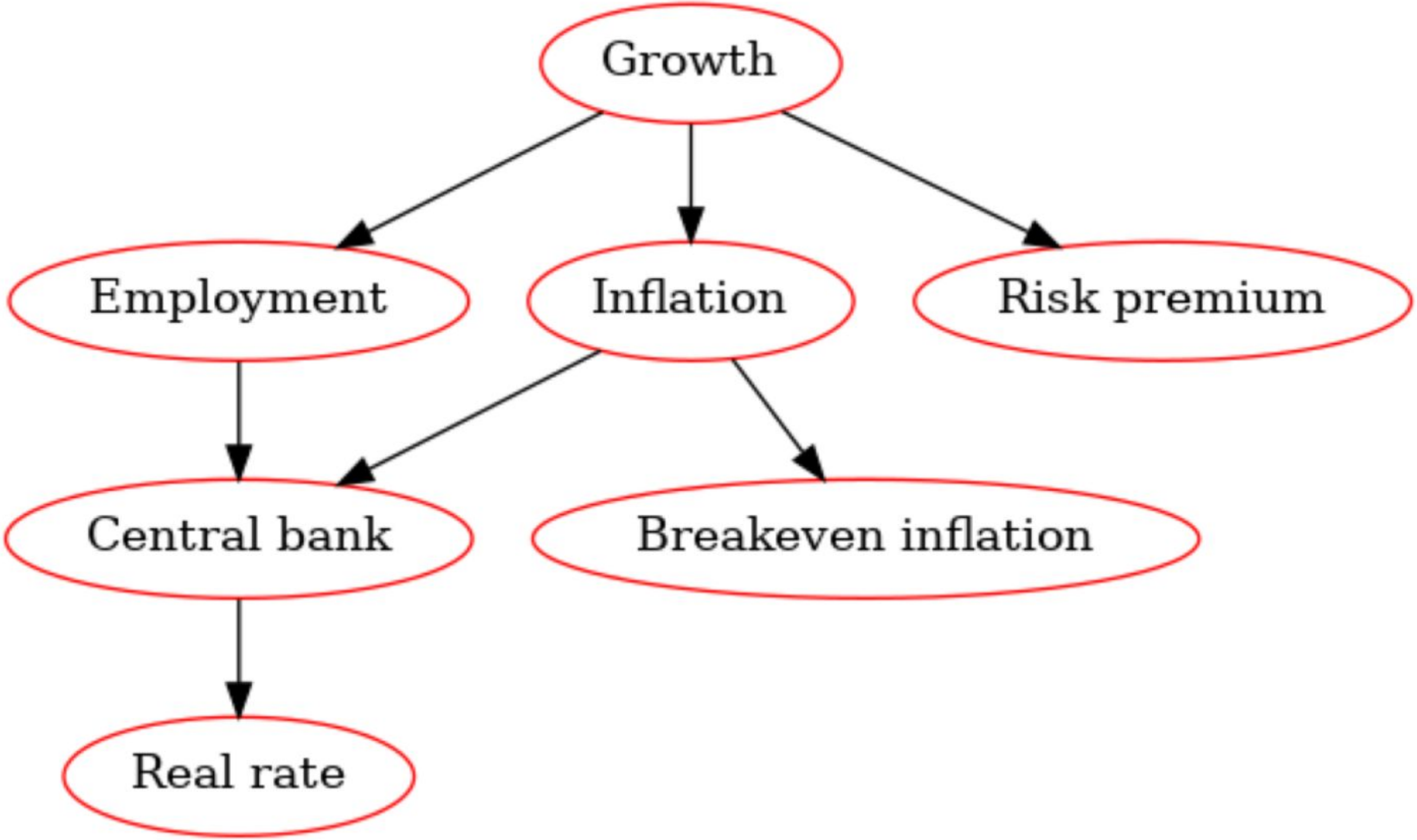


Sequential Entropy Pooling

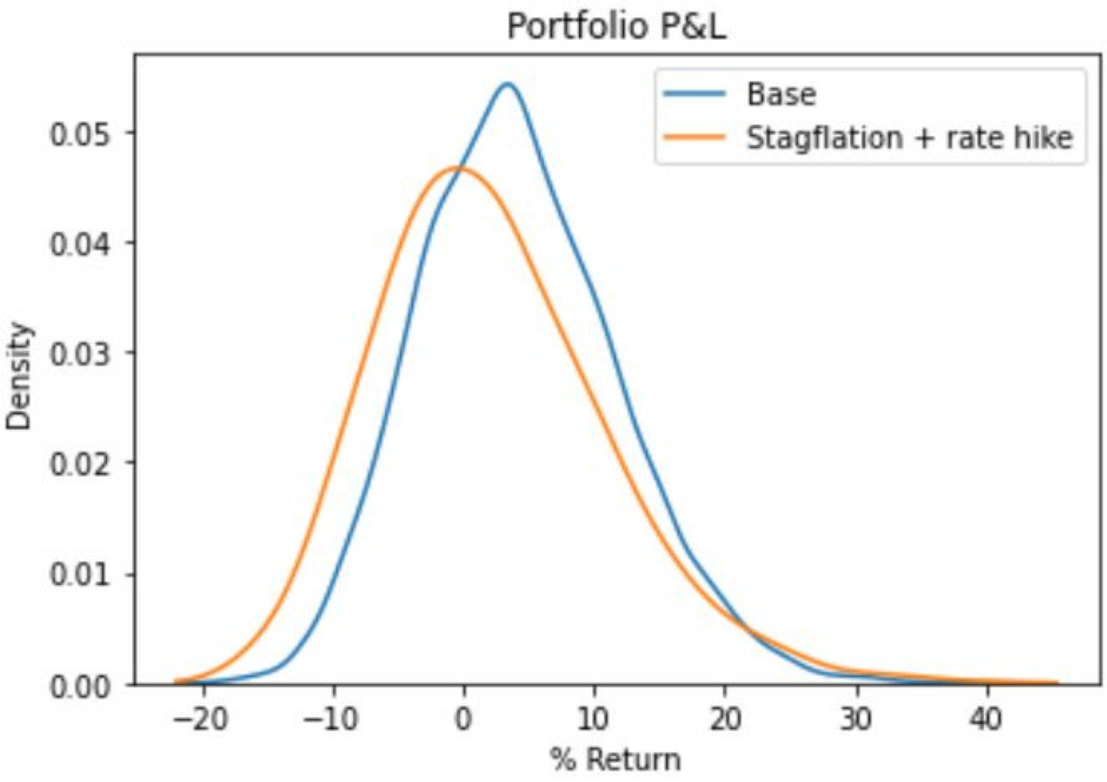


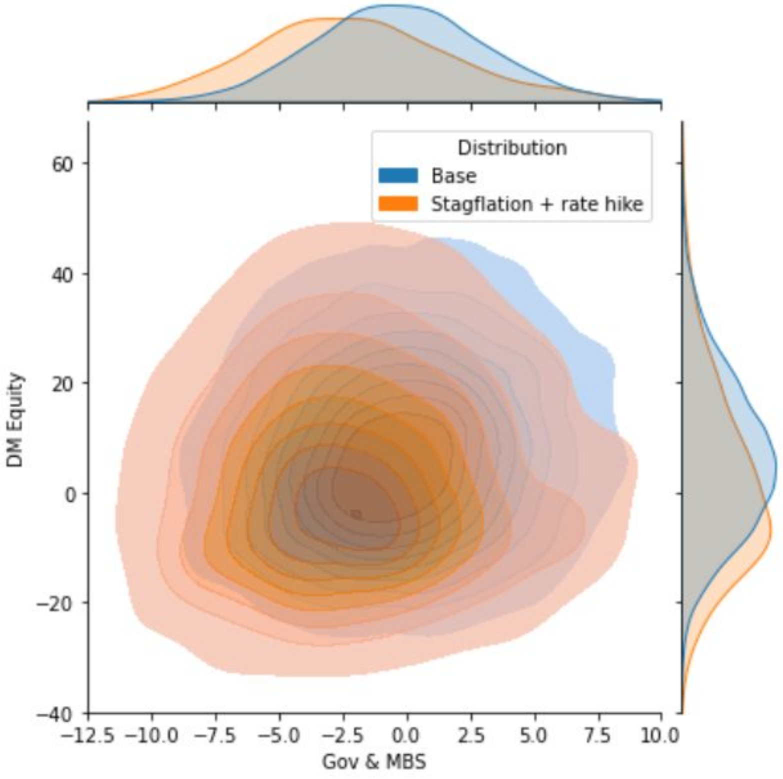


Causal and predictive views and stress-testing

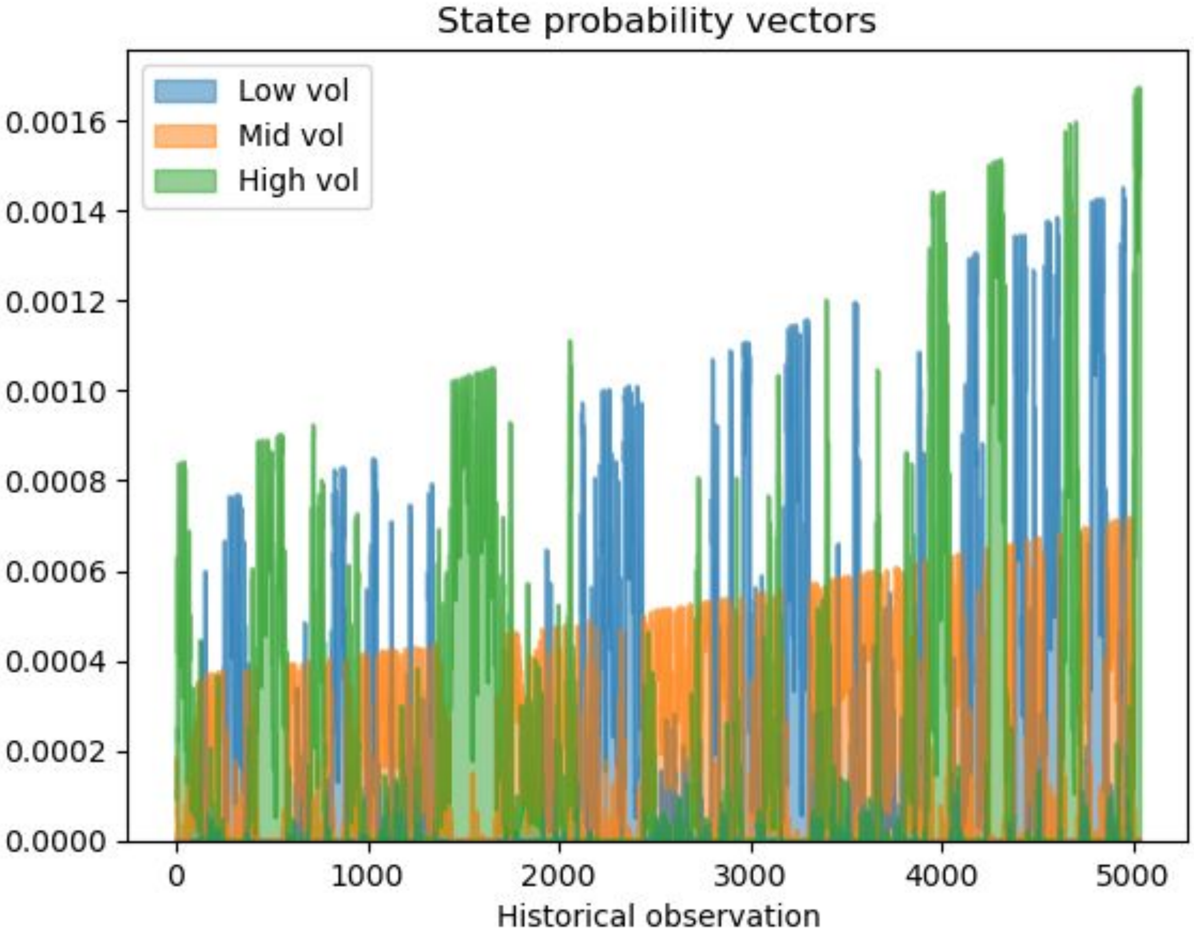


Causal and predictive views and stress-testing





Time- and State-Dependent Resampling



Portfolio optimization

$$e^{\star} = \text{mean-CVaR}\left(e, R, q, \mathcal{E}\right)$$

$$\text{s.t.} \quad v^{T} e + TC\left(\Delta e\right) = 1$$

$$e \in \mathcal{E} \subseteq \mathbb{R}^{I}$$

VS

$$w^{\star} = \text{mean-variance}\left(w, \mu, \Sigma, \mathcal{W}\right)$$

$$\text{s.t.} \quad \sum_{i=1}^{I} w_i = 1$$

$$w \in \mathcal{W} \subseteq \mathbb{R}^{I}$$

Conclusions

- All methods operate on fully general distributions and their associated joint probability vectors.
- Not trivial to generate realistic market scenarios, but possible to do much better than the normal distribution.
- Stylized market facts presented in Chapter 2.
- Market simulation framework and methods in Chapter 3.

Setting up Python

- Miniconda for package and environment management, ideally through Windows Subsystem for Linux (WSL).
- VS Code as the editor (Jupyter Notebook extension).
- Git and GitHub for version control.
- GitHub course repository: https://github.com/fortitudo-tech/pcrm-book