Improving random sampling in Python

scipy.stats.sampling and scipy.stats.qmc

Tirth Patel
Christoph Baumgarten
Matt Haberland

Pamphile T. Roy

tupui *PamphileRoy*

Quansight

SciPy

Statistics, optimization, interpolation, integration, *

Random numbers you said?

>>> Draw from a set: variable

... Numbers

... Things/Variables are dimensions

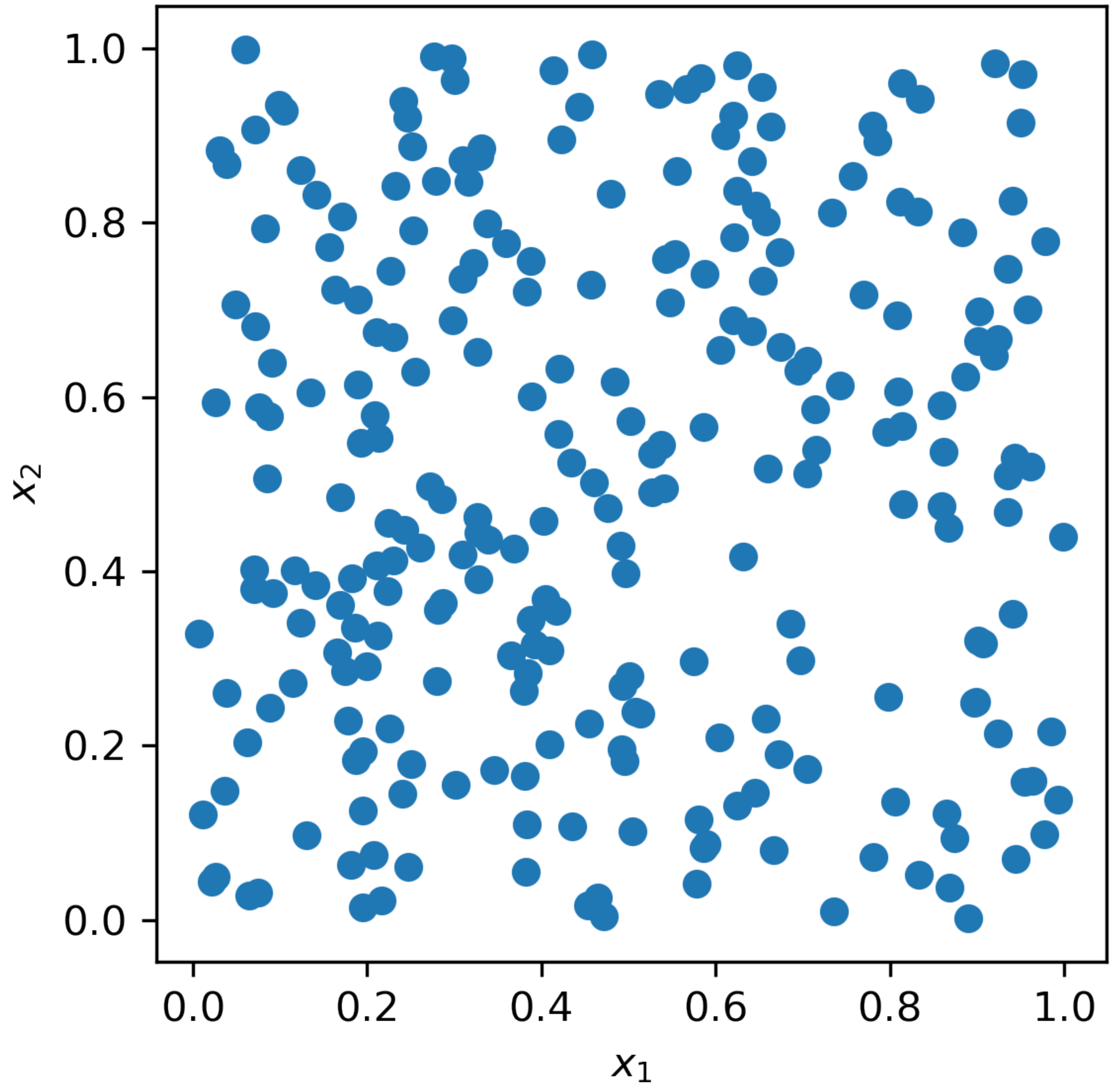
... Draw/samples are independent

```
>>> rng = np.random.default_rng()
>>> x = rng.random(1_000)

>>> x = rng.random(size=(1_000, 10))
```

Random numbers you said?

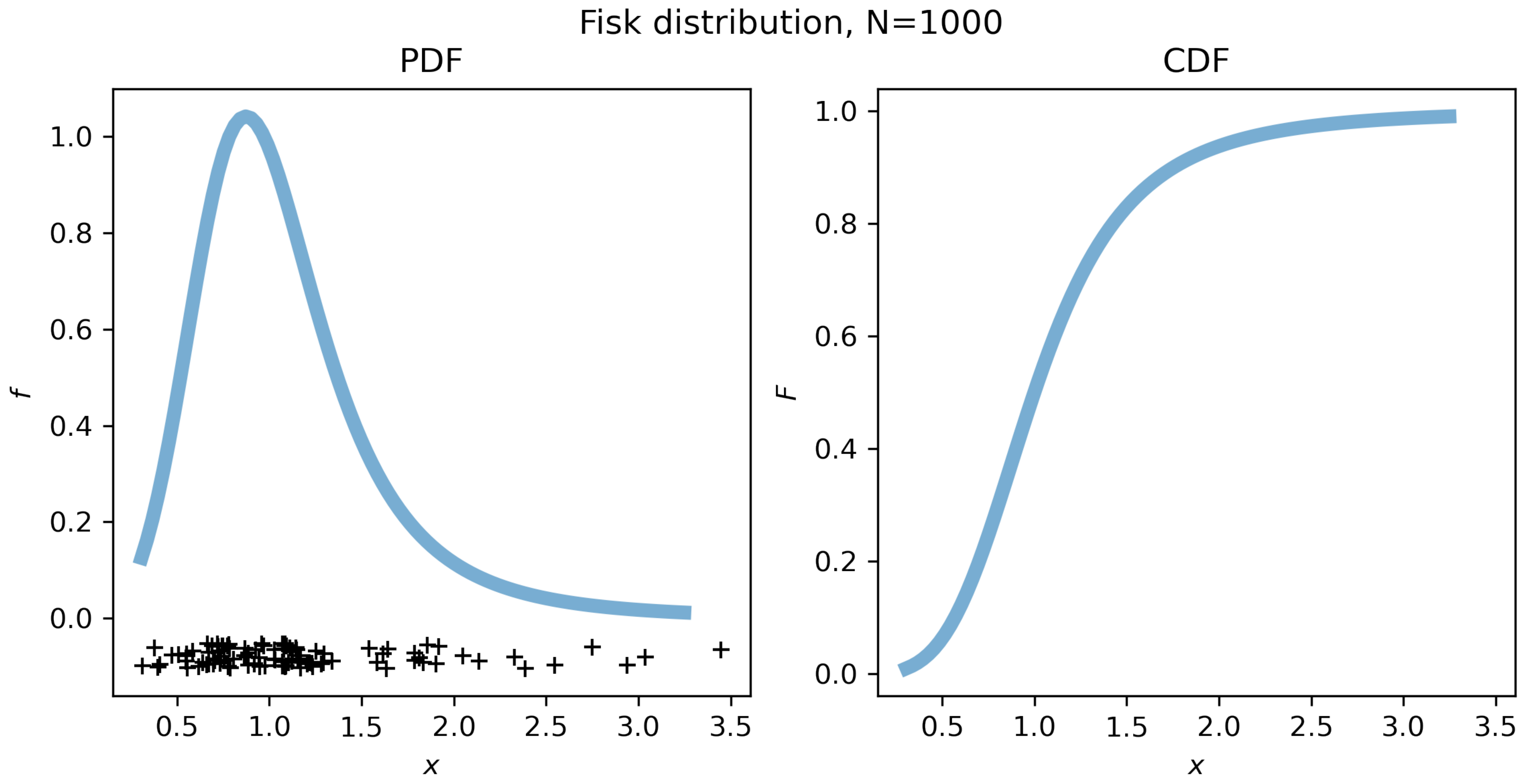
$$\mathbf{x} = (x_1, \ldots, x_n)$$

$$x \sim \mathcal{U}(0,1)$$



Distribution sampling is easy

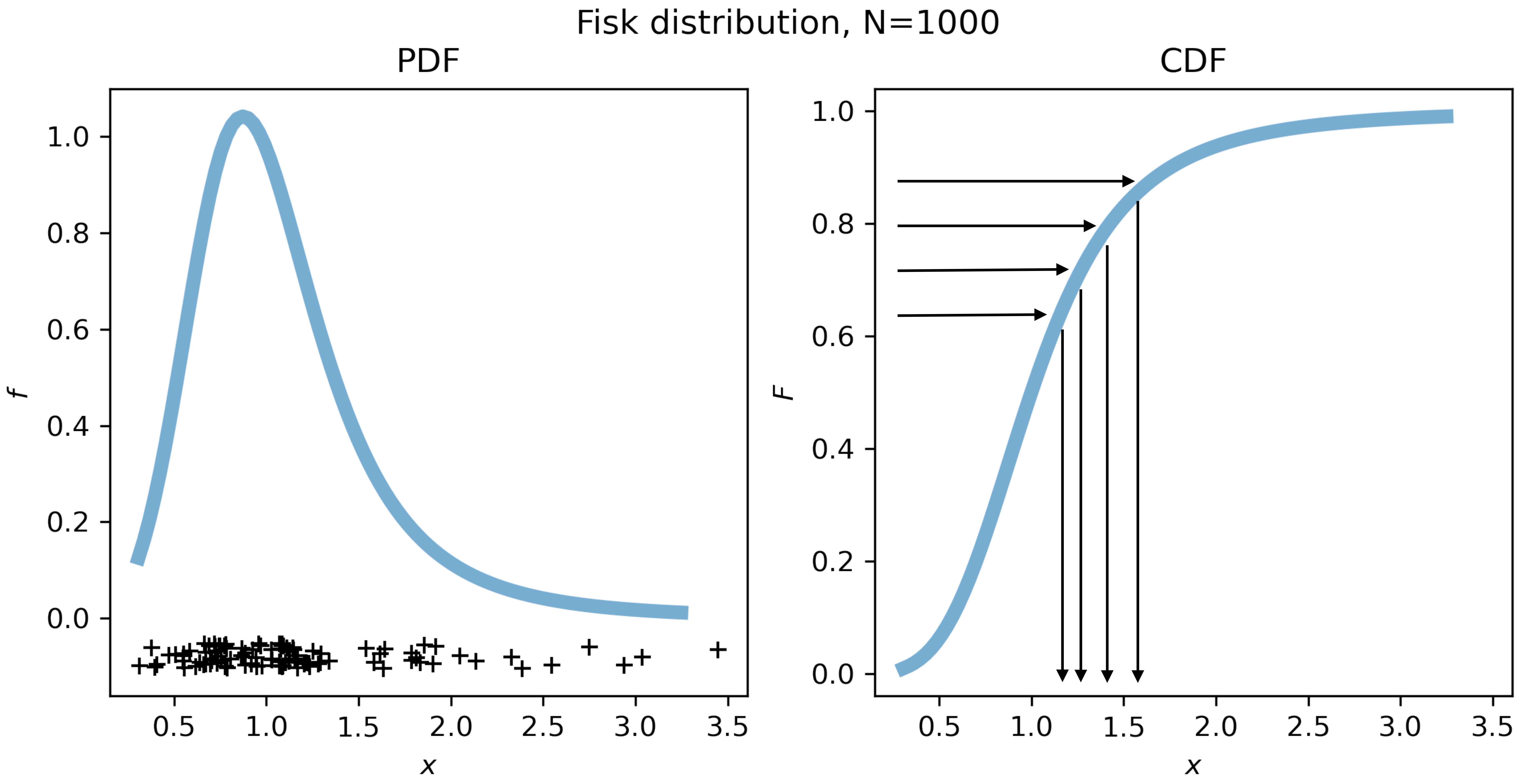
```
>>> x = U~0, 1()

>>> PDF = f(x)

>>> CDF = F(x)
```



Distribution sampling is easy

Inverse CDF: $F^{-1}(x') = x$

```
>>> rng = np.random.default_rng()
>>> x_ = rng.random(1_000)
>>> x = dist.ppf(x_)
```



But it’s hard

>>> What is this PPF function?

>>> What to do if there is no direct formula!?

>>> from scipy.stats import sampling

>>> Blackbox methods

>>> C library UNU.RAN

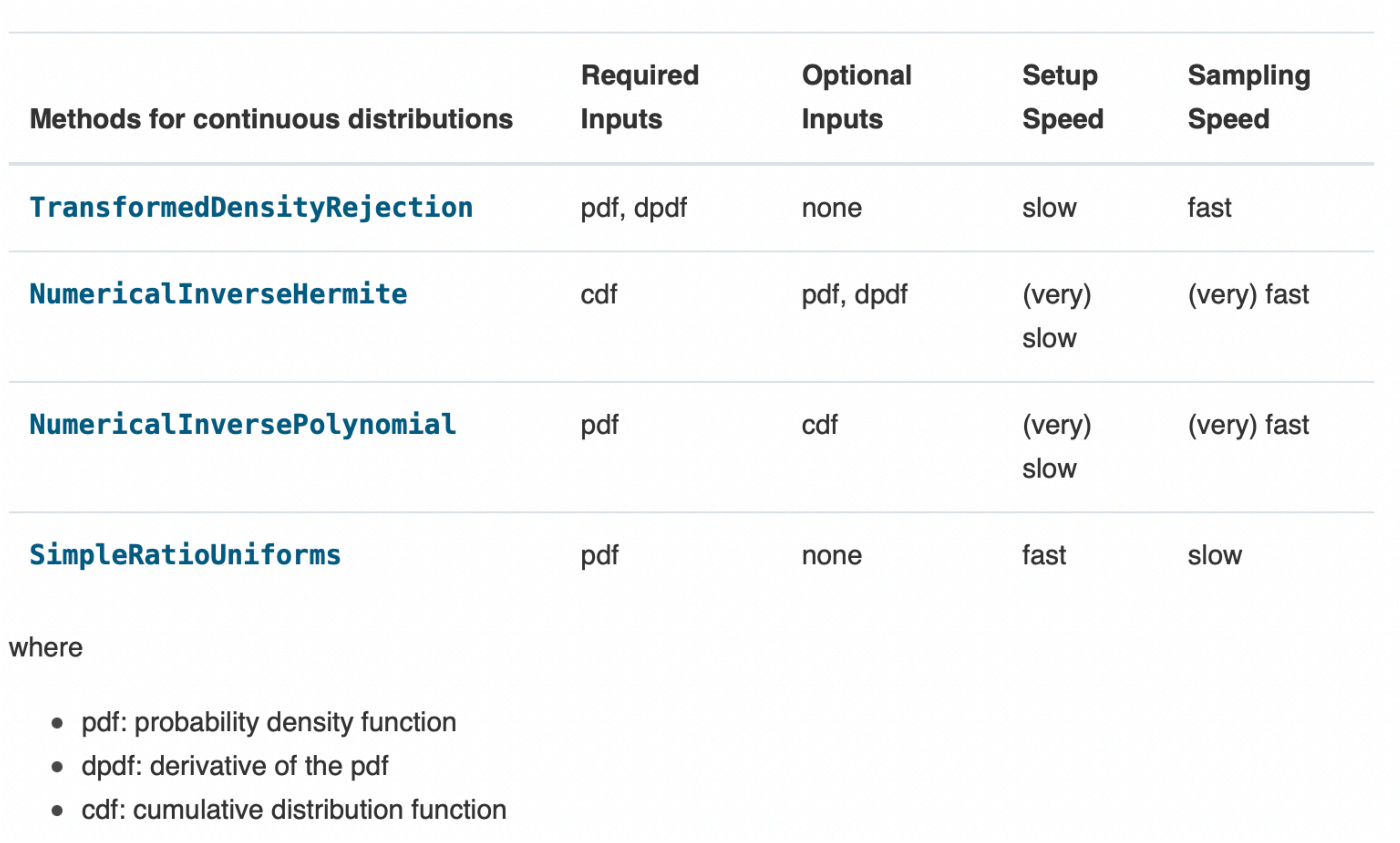
>>> Generate variates from non-standard distributions

>>> Inversion methods are relevant for QMC

>>> "*Automatic random variate generation in Python*"
in the conference proceedings

https://docs.scipy.org/doc/scipy/reference/stats.sampling.html

Sampling Continuous distributions



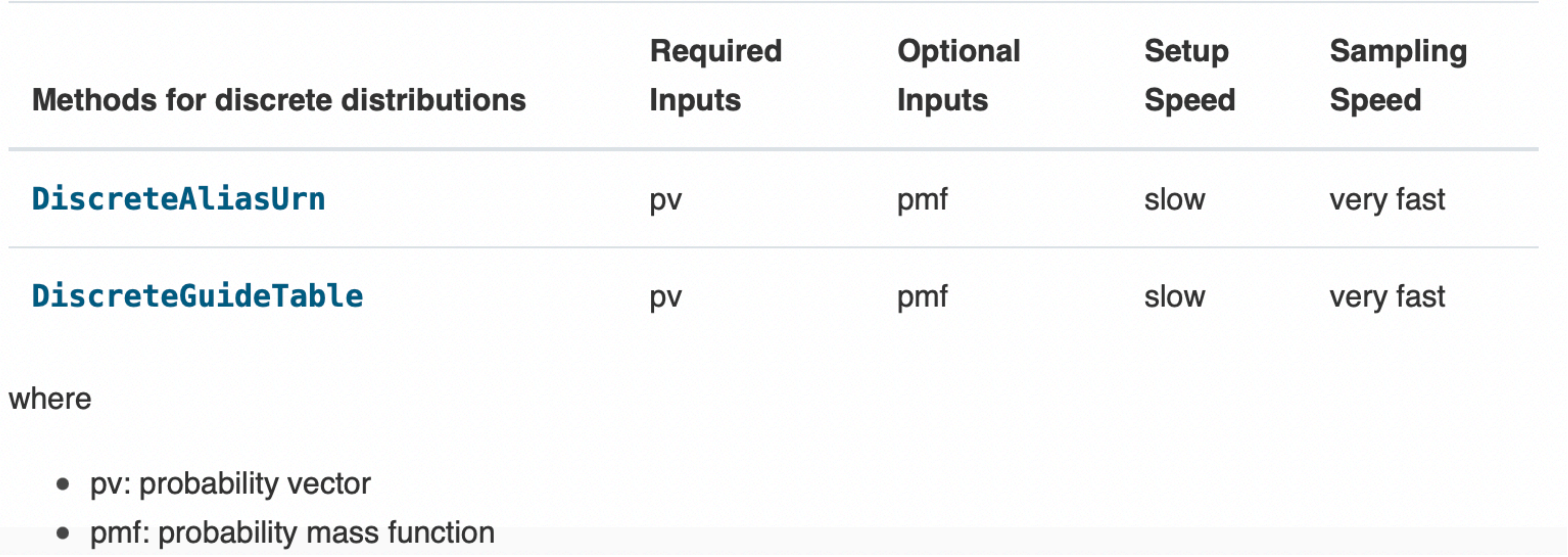
Methods for continuous distributions	Required Inputs	Optional Inputs	Setup Speed	Sampling Speed
`TransformedDensityRejection`	pdf, dpdf	none	slow	fast
`NumericalInverseHermite`	cdf	pdf, dpdf	(very) slow	(very) fast
`NumericalInversePolynomial`	pdf	cdf	(very) slow	(very) fast
`SimpleRatioUniforms`	pdf	none	fast	slow

where

- pdf: probability density function
- dpdf: derivative of the pdf
- cdf: cumulative distribution function

https://docs.scipy.org/doc/scipy/reference/stats.sampling.html

And Discrete distributions



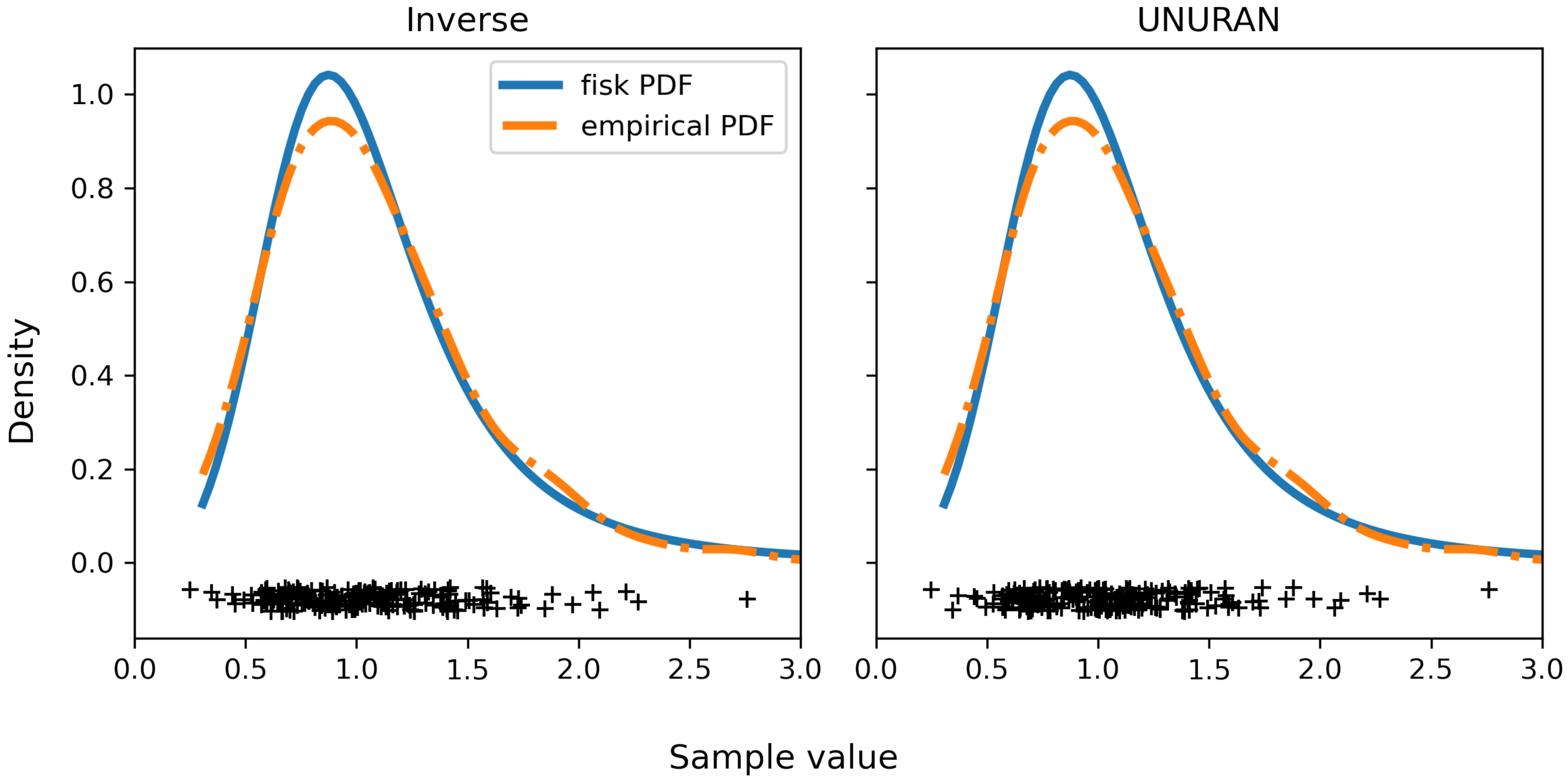
Methods for discrete distributions	Required Inputs	Optional Inputs	Setup Speed	Sampling Speed
`DiscreteAliasUrn`	pv	pmf	slow	very fast
`DiscreteGuideTable`	pv	pmf	slow	very fast

where

- pv: probability vector
- pmf: probability mass function

https://docs.scipy.org/doc/scipy/reference/stats.sampling.html

Sample various distributions



```
>>> import scipy.stats as stats
>>> dist = stats.fisk(c=3.9)
>>> rng_dist = stats.sampling.NumericalInverseHermite(dist)
>>> sample_qmc = rng_dist.rvs(128)
```

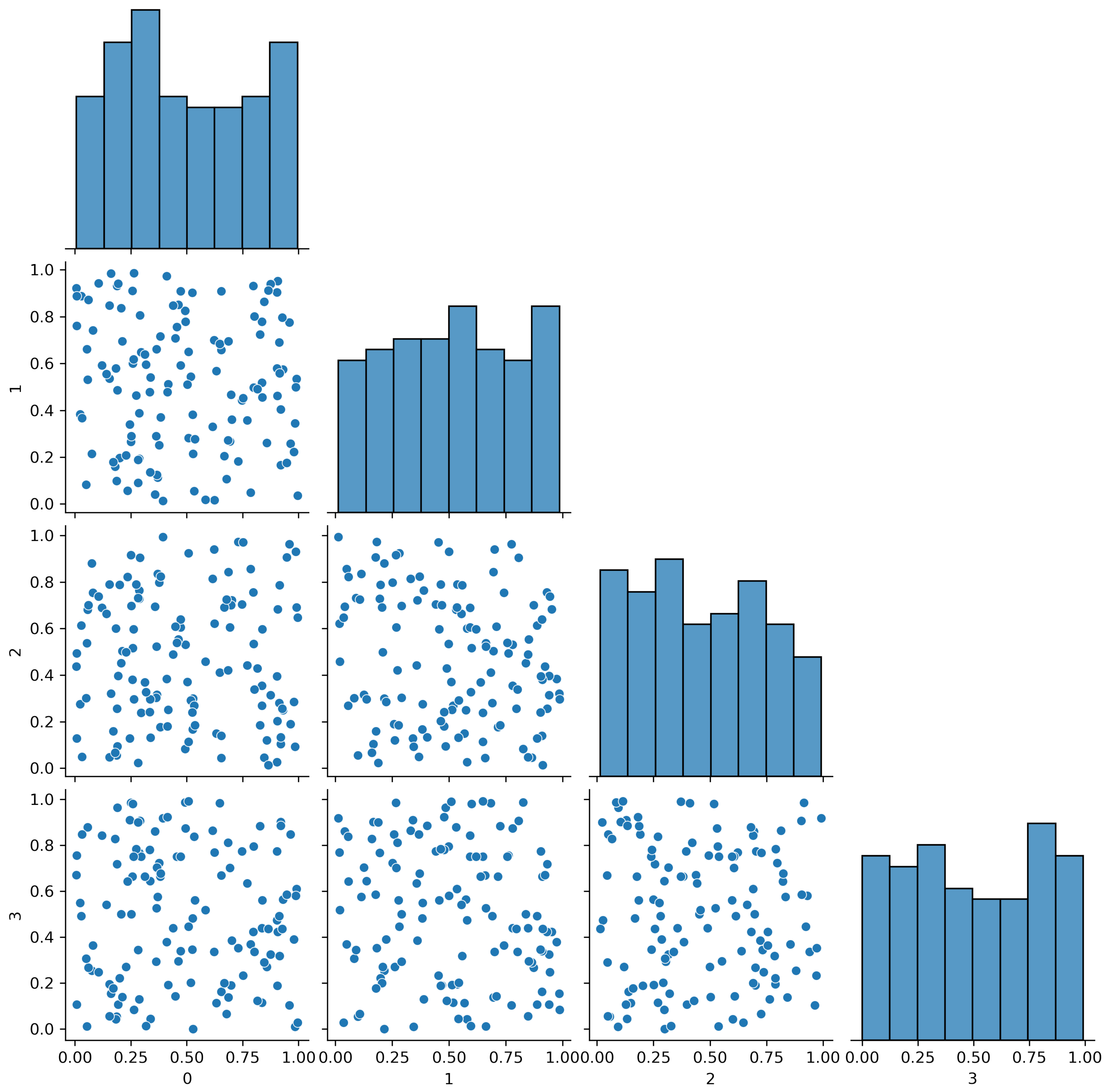
https://docs.scipy.org/doc/scipy/tutorial/stats/sampling.html

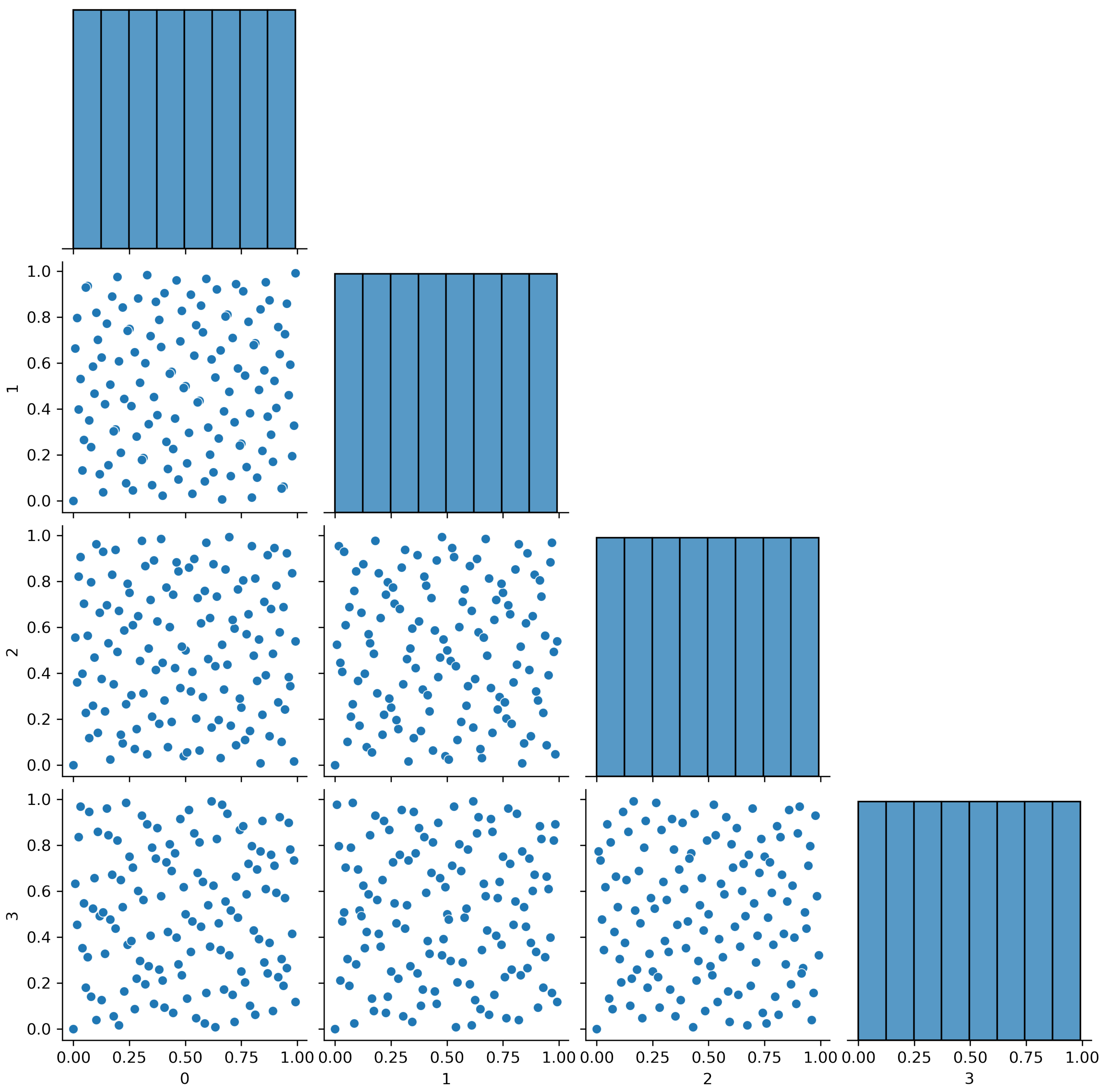
What is Quasi-Monte Carlo?

- Deterministic methods
- Not **i**ndependent and **i**dentically **d**istributed (IID)
- Better space filling
- Better convergence

https://blog.scientific-python.org/posts/scipy/qmc-basics

What is Quasi-Monte Carlo?





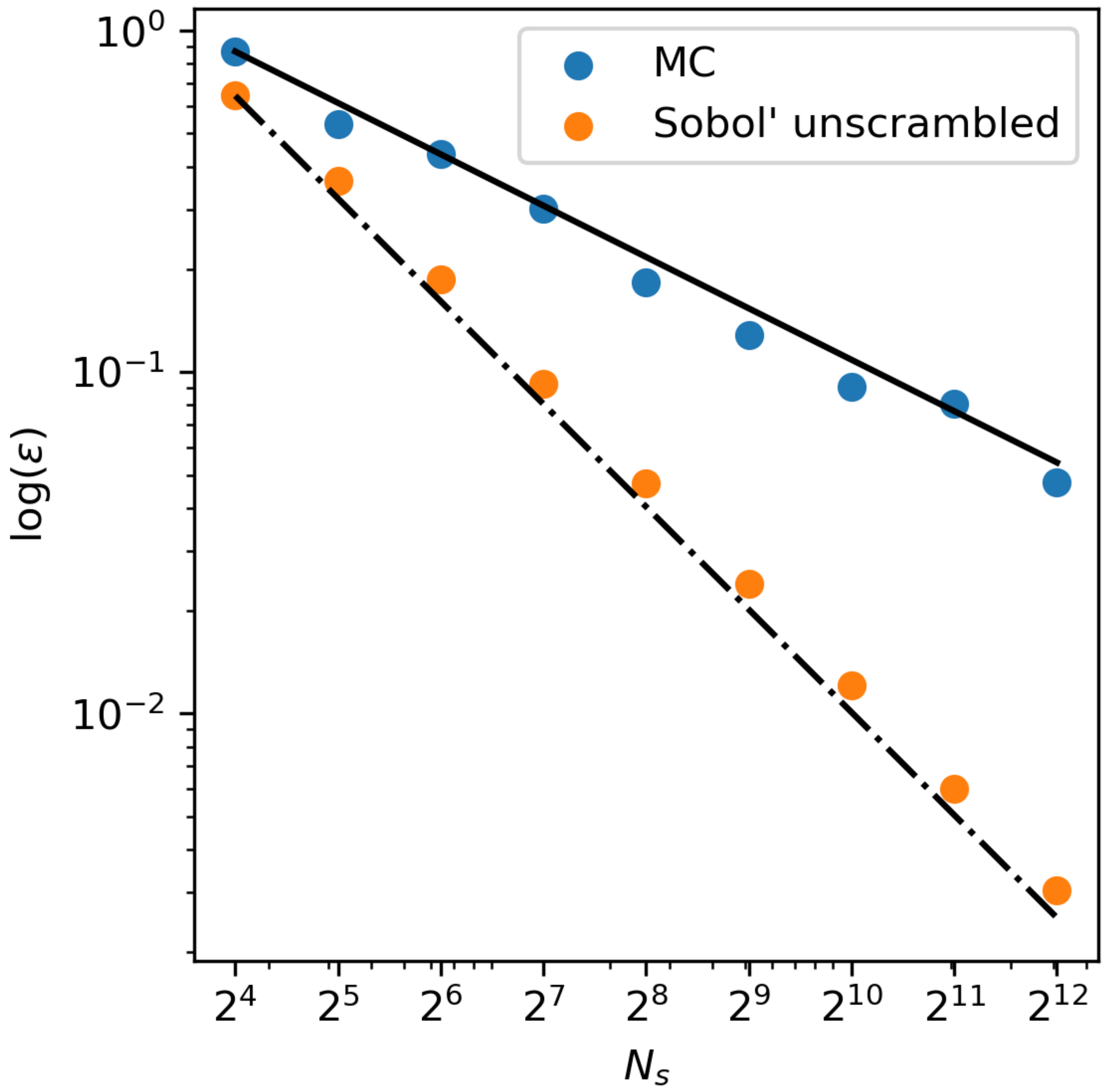
https://blog.scientific-python.org/posts/scipy/qmc-basics

What is Quasi-Monte Carlo?

$$f(\mathbf{x}) = \left(\sum_{j=1}^{5} x_j \right)^2$$

$$x_j \sim \mathcal{U}(0,1)$$

$$\mu = 5/3 + 5(5-1)/4$$



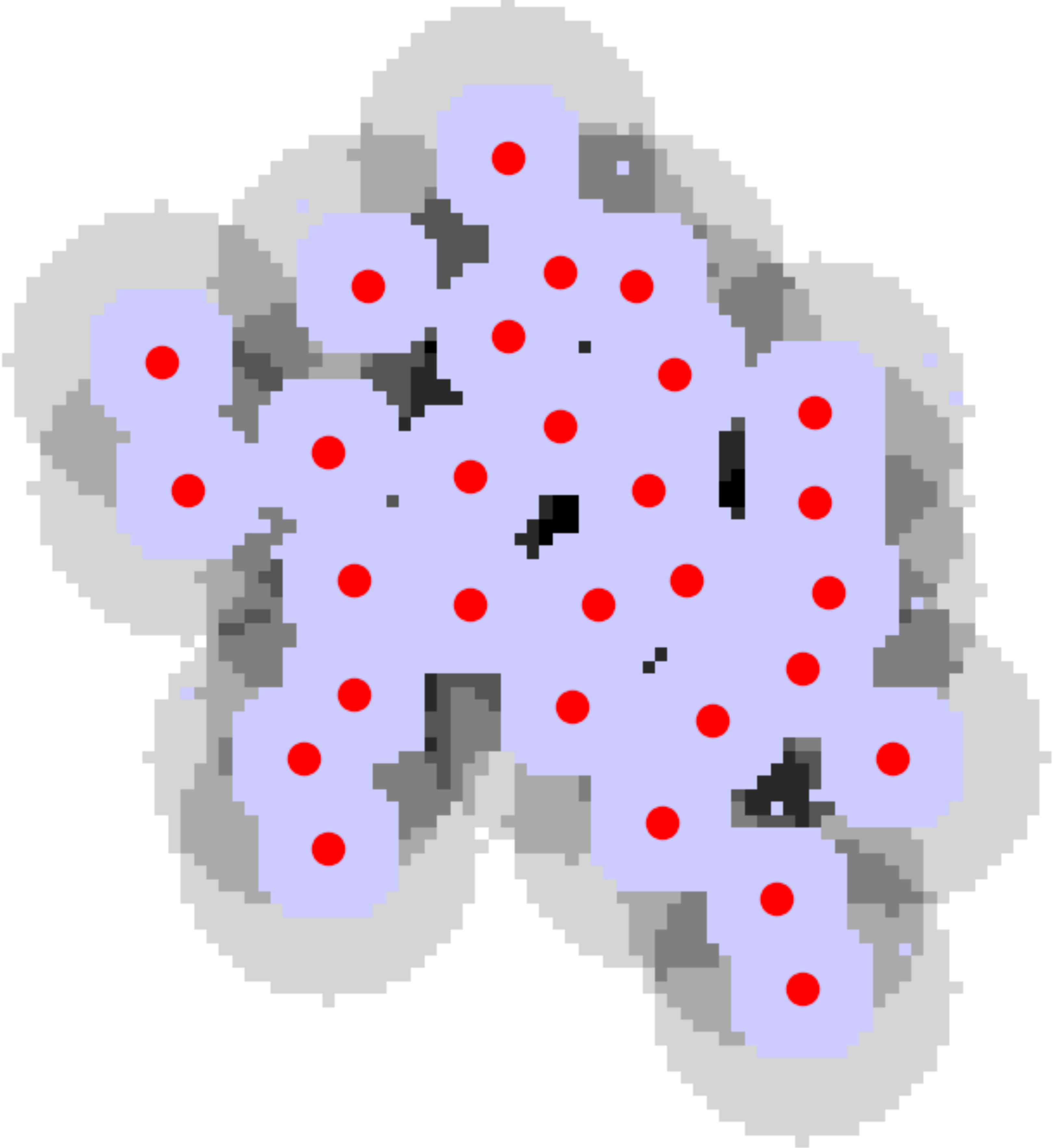
>>> Faster convergence integration

... o($n^{-1/2}$) to o(n^{-1})

https://blog.scientific-python.org/posts/scipy/qmc-basics

>>> from scipy.stats import qmc

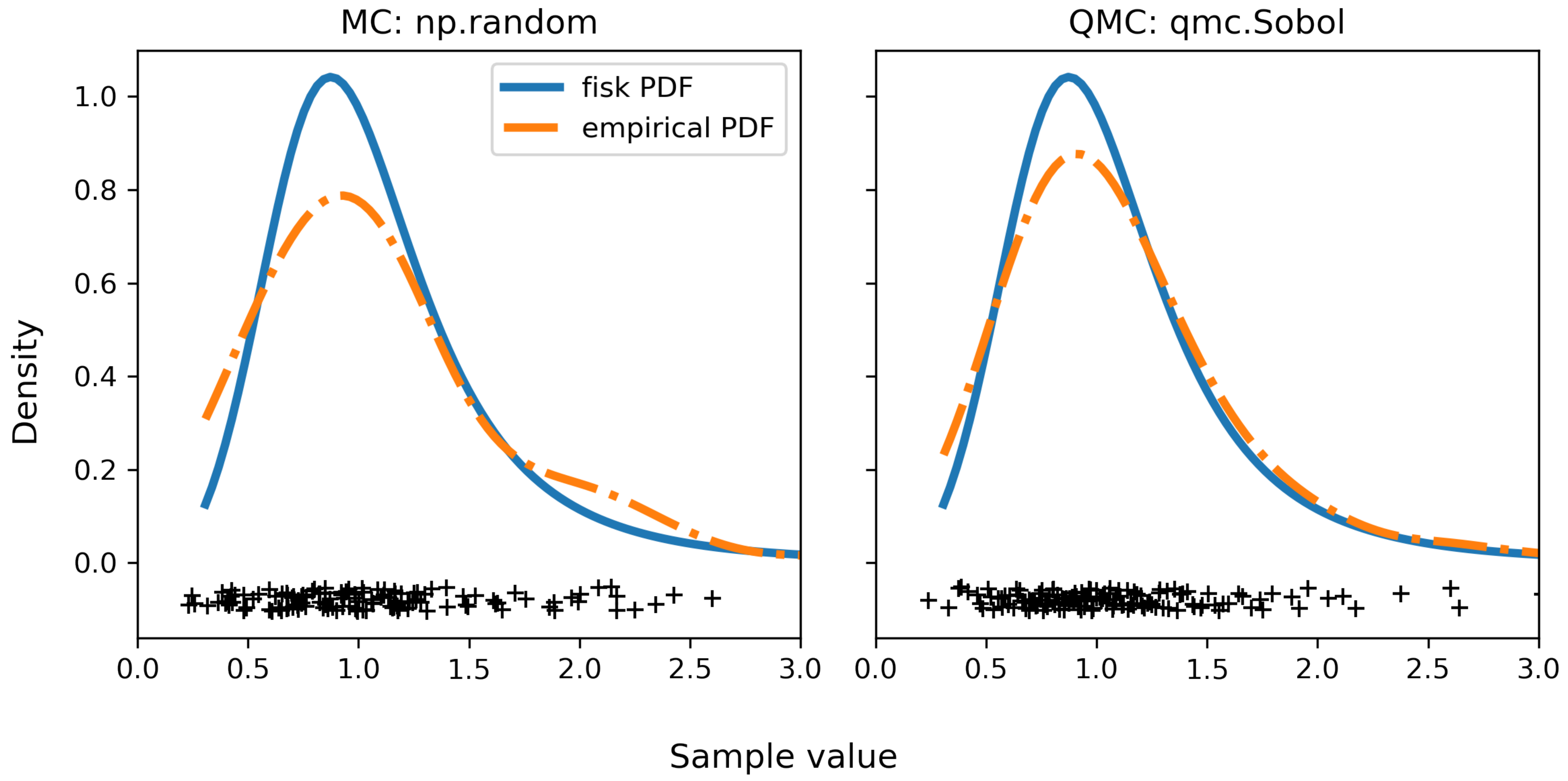
- >>> Sobol’
- >>> Halton
- >>> LHS and Orthogonal LHS
- >>> Poisson disk (new!)
- >>> Multinomial
- >>> Multivariate normal
- >>> Discrepancy



https://docs.scipy.org/doc/scipy/reference/stats.qmc.html

Sample various distributions with QMC

- >>> Easy API
- >>> sampling



```
>>> import scipy.stats as stats
>>> dist = stats.fisk(c=3.9)
>>> rng_dist = stats.sampling.NumericalInverseHermite(dist)
>>> sample_qmc = rng_dist.qrvs(128)
```

Do’s and don’ts

>>> Know where speed matters: sampling/init

>>> Use QMC!

>>> Don’t modify the sequences

>>> Don’t use a seed (ready to die on this hill)

Demo

https://gist.github.com/tupui/9c61591395da3a0b008e4be0ebb465d7

You help[ed] make it happen…

Issues, PR, reviews, discussions, issue management: *every contribution matters, even a star on the repository…* Join us!

scipy/scipy

Thank You!