Direct sampling & plots

Sample distribution

Every distribution in `m.dist` can be sampled **directly**, outside of a model, by passing `sample=True`. This is useful for prior predictive checks, simulating data, or quickly inspecting a distribution's shape.

```
from BayesForge import bf
m = bf()
# Draw 1000 samples from a standard normal
m.dist.normal(0, 1, shape=(4,), sample=True)
```

```
/home/sosa/work/3.12venv/lib/python3.13/site-packages/tqdm/auto.py:21: TqdmWarning: IProgress
  from .autonotebook import tqdm as notebook_tqdm
An NVIDIA GPU may be present on this machine, but a CUDA-enabled jaxlib is not installed. Fa
```

```
bf v 0.0.58 package loaded
jax.local_device_count 32
```

```
Array([ 0.70881297, -0.61375044, -1.00774708, -2.02585777], dtype=float64)
```

By using (`to_jax=False`), a direct sample is returned as a **`SampledData`** object — a thin wrapper around a JAX array. It behaves like a JAX array everywhere (arithmetic, indexing, `jnp.*` functions, `jit`/`vmap`/`grad`, use as a distribution parameter), and its `repr` reports the dtype so you can see the kind and precision at a glance:

The extra thing a `SampledData` gives you over a raw array is a set of **built-in plotting methods** and **built-in helper functions**.

Plotting helpers

Each plotting method is called directly on the sampled object. Most accept `interactive=True` (the default, **Plotly**) or `interactive=False` (**matplotlib / seaborn**):

The available methods depend on the **shape** of the sampled array.

1D-(n_samples,)

histogram

```
m.dist.normal(0, 1, shape=(1000,), sample=True, to_jax=False).hist()
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): text/html
```

kernel

```
m.dist.normal(0, 1, shape=(1000,), sample=True, to_jax=False).density()
```

```
Unable to display output for mime type(s): text/html
```

PPC

```
m.dist.normal(0,1,shape=(100,4),sample=True, to_jax=False).ppc_plot(m.dist.normal(0,1,shape=
```

```
Unable to display output for mime type(s): text/html
```

Time series — `(n_samples, n_time)`

Treat the second axis as time and summarise across draws.

```
m.dist.gaussian_random_walk(scale=0.5, num_steps=50, shape=(500,), sample=True, to_jax=False
```

```
Unable to display output for mime type(s): text/html
```

2D-(n_samples, n_variables)

histogram

```
m.dist.normal(0, 1, shape=(1000,4), sample=True, to_jax=False).hist()
```

```
Unable to display output for mime type(s): text/html
```

kernel

```
m.dist.normal(0, 1, shape=(1000,4), sample=True, to_jax=False).density()
```

```
Unable to display output for mime type(s): text/html
```

boxplot

```
m.dist.normal(0, 1, shape=(1000,4), sample=True, to_jax=False).boxplot()
```

```
Unable to display output for mime type(s): text/html
```

violinplot

```
m.dist.normal(0, 1, shape=(1000,4), sample=True, to_jax=False).violinplot()
```

```
Unable to display output for mime type(s): text/html
```

pairplot

```
m.dist.normal(0, 1, shape=(1000,4), sample=True, to_jax=False).pairplot()
```

```
Unable to display output for mime type(s): text/html
```

corr_heatmap

```
m.dist.lkj_cholesky(4, 2, name = 'a', sample = True, to_jax=False).corr_heatmap()
```

```
Unable to display output for mime type(s): text/html
```

traceplot

```
m.dist.normal(0, 1, shape=(1000,4), sample=True, to_jax=False).traceplot()
```

```
Unable to display output for mime type(s): text/html
```

surface_3d

Computes a 2D Kernel Density Estimate (KDE) using the first two variables. It plots a 3D surface where the X and Y axes represent the variables and the Z axis represents the estimated probability density.

```
m.dist.normal(0, 1, shape=(1000,4), sample=True, to_jax=False).surface_3d()
```

```
Unable to display output for mime type(s): text/html
```

scatter3d

Takes the first three variables (columns) and plots them directly as points in a 3D space (X, Y, and Z axes). Requires the array to have at least 3 variables (columns).

```
m.dist.normal(0, 1, shape=(1000,4), sample=True, to_jax=False).scatter3d()
```

```
Unable to display output for mime type(s): text/html
```

Posterior predictive

Compare draws against observed data.

3D-(n_samples, n_groups, n_times)

3D scatter — `(n_samples, >=3)`

```
m.dist.normal(0,1, name = 'a', shape=(100,4, 5), sample = True, to_jax=False).hist()
```

```
Unable to display output for mime type(s): text/html
```

Non-plot helper

In addition to plot helper functions, there are several other helper functions that can be used to compute summaries or transform the sampled array. For reduction operations (like `mean`, `std`, `sum`), you can pass an `axis` argument to reduce along specific dimensions. For example, passing `axis=0` will reduce across the sample dimension.

mean

```
m.dist.normal(0,1, name = 'a', shape=(100,4), sample = True, to_jax=False).mean(axis=0)
```

```
SampledData([ 0.09930412,  0.08486471,  0.16269986, -0.04464273], dtype=float64)
```

std

```
m.dist.normal(0,1, name = 'a', shape=(100,4), sample = True, to_jax=False).std()
```

```
Array(0.99340247, dtype=float64)
```

variance

```
m.dist.normal(0,1, name = 'a', shape=(100,4), sample = True, to_jax=False).var()
```

```
Array(1.00690316, dtype=float64)
```

hdi

```
m.dist.normal(0,1, name = 'a', shape=(100,4), sample = True, to_jax=False).hdi(cred_mass=0.95
```

```
(Array([0.06657551, 0.10564758, 0.12776344, 0.31720734], dtype=float64),
 Array([-1.02996427, -0.8243636 , -0.72078736,  0.07104139], dtype=float64))
```

sum

```
m.dist.normal(0,1, name = 'a', shape=(100,4), sample = True, to_jax=False).sum()
```

```
Array(9.04652727, dtype=float64)
```

prod

```
m.dist.normal(0,1, name = 'a', shape=(100,4), sample = True, to_jax=False).prod()
```

```
Array(-3.56643713e-113, dtype=float64)
```

min

```
m.dist.normal(0,1, name = 'a', shape=(100,4), sample = True, to_jax=False).min()
```

```
Array(-3.64392909, dtype=float64)
```

max

```
m.dist.normal(0,1, name = 'a', shape=(100,4, 5), sample = True, to_jax=False).max()
```

```
Array(4.12758551, dtype=float64)
```

argmin

```
m.dist.normal(0,1, name = 'a', shape=(100,4), sample = True, to_jax=False).argmin()
```

```
Array(294, dtype=int64)
```

argmax

```
m.dist.normal(0,1, name = 'a', shape=(100,4), sample = True, to_jax=False).argmax()
```

```
Array(253, dtype=int64)
```

all

```
m.dist.normal(0,1, name = 'a', shape=(100,4, 5), sample = True, to_jax=False).all()
```

```
Array(True, dtype=bool)
```

any

```
m.dist.normal(0,1, name = 'a', shape=(100,4), sample = True, to_jax=False).any()
```

```
Array(True, dtype=bool)
```

cumsum

```
m.dist.normal(0,1, name = 'a', shape=(100,4), sample = True, to_jax=False).cumsum()
```

```
SampledData([-1.46002866, -0.04446573, -1.75294629, -2.36816525, -1.02214756,
       -0.80975621, -1.63783133, -1.65558972, -2.05676285, -2.09916228,
       -3.19452752, -3.7724002 , -2.95926821, -3.81926984, -2.71335542,
       -2.53300772, -2.07195281, -1.29237334, -0.34173531,  0.64443439,
        0.6092218 , -0.03548849,  0.11714811,  0.98955158,  0.92941958,
        2.45116977,  4.04269793,  3.71886159,  4.40338225,  3.64111511,
```

```
 4.12745848,  5.10347852,  3.85317929,  3.80698842,  3.13686069,
 4.65121342,  5.70530175,  4.84509284,  6.40925436,  5.89950044,
 5.134761  ,  5.48916246,  5.64019832,  7.60011966,  5.58003842,
 6.75674319,  6.25898973,  6.99592801,  7.82088954,  7.47487593,
 7.10209033,  6.63958381,  7.1113472 ,  6.5683649 ,  6.62944573,
 7.20968423,  6.66356258,  7.92839984,  9.58269393,  9.06172946,
 9.87359203, 10.02737628, 11.70203805, 10.39454393, 10.43283864,
10.93376547, 12.30231232, 12.33708499, 11.71735585, 10.88658881,
11.90282632, 11.96704935, 12.71539476, 13.59820111, 13.0438412 ,
12.6910104 , 13.06052158, 13.50296007, 13.53056581, 12.06401904,
11.59454715,  9.42695676,  9.84172672,  8.76215852,  8.20683638,
 7.5029634 ,  9.23809366, 11.88786153, 12.28829636, 12.79183918,
11.38601376, 12.41847087, 12.31373403, 13.28193288, 14.44999656,
14.92977007, 14.45759833, 13.74290437, 13.63377235, 15.23548842,
14.01826882, 13.98309945, 13.93120209, 13.4648141 , 14.09666945,
12.49839553, 13.10824329, 14.94341289, 15.18868667, 15.88376054,
15.99431544, 15.83583953, 15.78113614, 17.51331394, 17.8287734 ,
19.7374284 , 21.79986062, 21.35071024, 21.11839043, 20.44071585,
18.85749679, 18.17526265, 18.16566227, 17.5046218 , 17.17261256,
17.59626737, 17.7297758 , 18.50376408, 18.39174767, 16.63255047,
17.32839452, 16.59663571, 17.49090678, 17.61101317, 18.26486156,
19.43511451, 20.94933181, 21.78014201, 22.1574559 , 21.83366737,
19.42077578, 17.78195744, 16.90957074, 17.63149568, 16.5280743 ,
16.8923547 , 15.87677878, 15.2293864 , 16.83752293, 16.33484766,
16.2929108 , 17.04031212, 15.24034651, 15.03913425, 13.73522492,
13.87868638, 13.5119597 , 14.10002521, 12.78571314, 13.94136662,
14.65717826, 14.04494581, 14.7849596 , 15.60461165, 15.94119016,
15.93030738, 16.07156789, 15.93305191, 15.89095397, 14.34741095,
15.00579695, 14.60732876, 17.07238458, 16.57606121, 16.47262739,
16.22304057, 16.82084801, 15.67472834, 14.3844936 , 13.78215542,
14.00025438, 13.01434352, 12.74099318, 12.29195467, 12.01142125,
10.40864864, 10.2706105 , 12.21946829, 11.87016257, 10.54597471,
11.04074302, 11.35791155, 12.02235698, 10.06511995, 12.13882774,
11.89001017, 11.96519965, 10.88622977,  9.88312361, 11.62105953,
12.59994276, 13.18460094, 13.14889531, 13.06736421, 12.02949359,
12.00630321, 12.29347639, 11.47974156, 11.04340432, 11.06211155,
10.86187858, 11.79803598, 11.18423398, 10.63367828, 11.05476323,
12.83198343, 11.57386269, 10.92196338,  9.50466963, 10.09510414,
10.18555736, 12.20900492, 11.77110168, 11.26864696, 12.52598983,
12.17745343, 12.19278319, 12.96883555, 13.43630321, 13.03802945,
12.0258599 , 11.28060286, 13.73812484, 13.89503684, 14.86155429,
16.01668997, 15.88302904, 16.3352373 , 15.39735383, 15.63346324,
15.96590295, 16.11025622, 17.03228857, 16.79959604, 16.53766691,
```

```
       17.38652281, 17.52350857, 17.29875074, 17.38929482, 17.39927667,
       16.21167857, 16.01642028, 15.80821942, 16.21213475, 15.86104525,
       15.40169784, 15.44649256, 16.12416354, 16.37731526, 16.00838647,
       16.49344785, 18.12879305, 17.19335661, 17.28037924, 17.07389948,
       17.10589698, 18.89423609, 19.29438172, 20.05283075, 19.47664048,
       19.41828571, 18.38064617, 18.16098196, 17.43446734, 19.0898184 ,
       18.24582905, 20.64403491, 20.08088955, 20.96472701, 20.6834048 ,
       21.93200916, 22.07539328, 21.20690814, 19.74266624, 19.37477423,
       18.83527938, 19.31350108, 19.19719181, 18.42582797, 18.84210575,
       18.8782343 , 19.89976338, 21.25769481, 18.30753208, 19.03569406,
       17.53210049, 16.78893614, 17.08457401, 15.6997456 , 14.27630631,
       14.11403743, 14.54669307, 14.99673133, 14.83240176, 15.4481094 ,
       15.57776915, 14.58727707, 14.51199693, 13.9808518 , 14.18521936,
       14.62671135, 15.65836699, 16.07081176, 16.27355032, 15.54690824,
       14.86266809, 14.12793098, 13.6324802 , 12.64095291, 11.79950545,
       12.71150459, 12.24758695, 12.25648259, 11.35776803, 10.91630645,
       10.06960991, 10.62612698, 11.06526152, 10.55590801, 12.04387366,
       10.63361908,  9.39937341, 10.79991389,  9.46553549,  9.3700526 ,
       10.13870828, 10.85392228, 11.66180943, 12.60450949, 11.90937743,
       11.64939328, 12.09440355,  9.21276849, 10.42377505, 10.37795256,
       10.46177141, 11.11895834, 11.37941732, 10.92496423, 10.30589146,
       11.34492625,  9.68625694,  7.87607432,  8.45134211,  7.35996659,
        6.11968685,  4.45720368,  2.67676837,  3.00792508,  3.64540626,
        3.6957532 ,  5.27233779,  4.35509382,  3.94799595,  4.02098564,
        3.70071602,  5.40038379,  5.53636558,  4.99704314,  3.09655311,
        4.42119309,  3.35058343,  3.7330445 ,  3.17797947,  5.20564099,
        4.25064507,  3.5598481 ,  4.13396103,  4.2190192 ,  2.7838448 ,
        3.45578924,  4.22866499,  4.72559167,  4.97743164,  5.80194599,
        5.38177424,  4.38982107,  5.12541063,  4.0471337 ,  2.74791761,
        2.75757571,  2.65064975,  4.78158479,  5.05511549,  5.3883384 ,
        6.38874408,  6.96119092,  7.79526396,  7.10046767,  6.81463531],      dtype=float64)
```

round

```
m.dist.normal(0,1, name = 'a', shape=(100,4, 5), sample = True, to_jax=False).round()
```

```
SampledData([[[ 2., -0., -0., -1., -1.],
        [-2., -1., -1., -1.,  2.],
        [-0.,  1.,  1., -1.,  1.],
        [-0.,  2., -1.,  0.,  0.]],
```

```
       [[-0., -2.,  1.,  0.,  2.],
        [-2., -0., -0., -0.,  0.],
        [ 1.,  1., -1., -1.,  1.],
        [ 1.,  1., -1.,  2., -2.]],

       [[-0.,  1., -2.,  0., -1.],
        [-1.,  0.,  2.,  1., -1.],
        [ 1., -2.,  1.,  0.,  1.],
        [ 1.,  0.,  1., -1., -0.]],

       ...,

       [[-0., -1.,  1.,  1.,  1.],
        [ 0., -0., -2.,  0., -1.],
        [ 0.,  0., -0.,  1., -0.],
        [ 0.,  0., -0.,  1., -1.]],

       [[-0.,  0.,  1., -0., -1.],
        [-0.,  2.,  0.,  1., -0.],
        [-2., -1., -1.,  0.,  1.],
        [-0., -2.,  1.,  1.,  1.]],

       [[ 0.,  1.,  1., -0., -0.],
        [ 1.,  2.,  1.,  1.,  1.],
        [-2.,  0., -1.,  1.,  1.],
        [ 2., -1.,  2.,  1., -1.]]], dtype=float64)
```

clip

```
m.dist.normal(0,1, name = 'a', shape=(100,4), sample = True, to_jax=False).clip(0, 1)
```

```
SampledData([[0.        , 1.        , 0.        , 0.        ],
       [0.67535797, 0.35376967, 0.19568032, 0.        ],
       [0.17518829, 0.        , 0.        , 0.31703263],
       [0.        , 0.        , 0.        , 0.        ],
       [0.        , 0.88104597, 0.25327077, 0.04193629],
       [0.        , 0.        , 1.        , 0.        ],
       [0.        , 1.        , 0.        , 0.        ],
       [0.        , 0.        , 0.25025799, 0.        ],
```

```
[0.        , 0.32166082, 0.63112891, 0.        ],
[0.        , 1.        , 1.        , 0.        ],
[1.        , 0.        , 1.        , 0.        ],
[1.        , 0.130894  , 1.        , 0.        ],
[0.07589962, 0.11540833, 0.        , 0.        ],
[0.        , 0.        , 0.        , 1.        ],
[1.        , 0.        , 1.        , 1.        ],
[0.        , 0.        , 0.        , 0.76725315],
[1.        , 0.81095748, 0.        , 0.52165367],
[0.        , 0.        , 0.40443199, 0.        ],
[0.91804061, 0.        , 0.        , 0.        ],
[0.58891829, 0.68867294, 1.        , 0.00844513],
[0.        , 0.02452011, 0.83718367, 0.10938338],
[0.        , 0.72244682, 0.        , 0.61844118],
[0.        , 0.60385552, 0.22882586, 0.        ],
[0.        , 0.85234651, 0.        , 0.        ],
[0.        , 0.58817265, 0.27826706, 0.82134955],
[0.21143582, 0.        , 0.39308008, 0.        ],
[0.        , 0.        , 0.46949572, 0.04327362],
[1.        , 0.        , 0.        , 0.38003418],
[0.        , 1.        , 0.25636326, 0.        ],
[0.        , 0.        , 0.44252596, 1.        ],
[0.        , 0.        , 0.        , 0.        ],
[0.        , 0.        , 1.        , 0.        ],
[0.        , 0.73867083, 0.04290433, 0.        ],
[0.13846954, 0.        , 0.57526245, 0.81841394],
[0.        , 0.        , 0.        , 1.        ],
[0.00802994, 0.46734415, 0.        , 0.17117431],
[0.        , 0.15194107, 0.47152285, 0.        ],
[0.        , 0.44336802, 1.        , 0.        ],
[1.        , 0.65585617, 0.78777288, 1.        ],
[0.29065874, 0.32747577, 1.        , 0.1887572 ],
[0.        , 0.16279199, 0.        , 0.55197005],
[0.        , 0.19724885, 0.7006229 , 1.        ],
[0.25909593, 0.22069029, 0.        , 0.        ],
[0.        , 0.91718275, 0.        , 0.53265759],
[0.        , 0.        , 0.        , 0.37015394],
[0.        , 1.        , 0.        , 0.21280144],
[0.24450803, 0.        , 0.        , 0.18627674],
[0.        , 0.        , 1.        , 0.        ],
[0.13275671, 0.        , 0.        , 0.33033705],
[1.        , 0.52351927, 0.        , 0.        ],
[0.38883989, 0.78325433, 0.64735072, 0.        ],
```

```
[0.        , 0.12803434, 0.97051155, 0.88119011],
[1.        , 0.        , 0.29392917, 0.        ],
[0.        , 0.        , 1.        , 0.        ],
[0.        , 0.        , 0.        , 1.        ],
[0.025024  , 0.52837996, 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        ],
[0.8412696 , 0.2723172 , 0.        , 0.        ],
[0.        , 0.15547122, 0.37159804, 0.33926159],
[0.52393899, 0.        , 0.        , 1.        ],
[0.        , 0.        , 0.52068962, 0.97304292],
[0.63068672, 1.        , 0.        , 0.56299802],
[1.        , 0.        , 0.        , 0.03135008],
[0.52897184, 0.        , 0.        , 1.        ],
[0.        , 0.        , 0.00293918, 0.        ],
[0.        , 0.        , 0.        , 0.65499142],
[0.        , 0.        , 0.        , 0.        ],
[1.        , 0.12537859, 0.90012983, 0.85948013],
[0.        , 0.02605914, 0.02140555, 0.3197932 ],
[0.06302727, 0.        , 1.        , 0.04415205],
[0.49757692, 0.        , 0.74108033, 1.        ],
[1.        , 0.        , 0.27808431, 0.36569515],
[0.2373126 , 0.        , 0.4541104 , 0.38018981],
[0.79811514, 0.        , 0.73292756, 0.62361505],
[0.        , 0.        , 0.        , 0.        ],
[0.60838075, 0.10969173, 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        ],
[0.        , 0.22397362, 0.        , 0.        ],
[0.45583614, 0.34928225, 1.        , 0.18976578],
[0.98101921, 0.        , 0.        , 0.6730398 ],
[0.42740003, 1.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        ],
[0.        , 1.        , 0.        , 0.        ],
[1.        , 1.        , 0.51389026, 0.        ],
[0.15523086, 0.        , 0.78890371, 0.        ],
[0.26059659, 0.        , 0.        , 0.        ],
[0.63756407, 0.93647535, 0.08123414, 0.        ],
[1.        , 0.        , 0.82387463, 1.        ],
[0.65603279, 0.91734372, 0.4057906 , 1.        ],
[0.93576065, 0.66199442, 0.        , 0.18208871],
[0.        , 0.87610845, 0.17505323, 0.56311619],
[0.39511712, 0.81699978, 1.        , 0.87412812],
[0.62369844, 0.        , 0.94093873, 0.        ],
[0.        , 0.1323173 , 0.        , 0.        ],
```

```
       [0.        , 0.        , 0.        , 1.        ],
       [0.56934061, 0.02416456, 0.37577079, 0.        ],
       [1.        , 0.        , 0.36079418, 1.        ],
       [0.        , 0.        , 0.        , 0.        ],
       [0.        , 1.        , 0.        , 0.09782381],
       [1.        , 0.26934573, 1.        , 0.        ]], dtype=float64)
```

ptp

```
m.dist.normal(0,1, name = 'a', shape=(100,4), sample = True, to_jax=False).ptp()
```

```
Array(5.89335262, dtype=float64)
```

sort

```
m.dist.normal(0,1, name = 'a', shape=(100,4), sample = True, to_jax=False).sort()
```

```
SampledData([[-1.07345528e+00, -9.72115343e-01,  9.19362447e-01,
         9.94440605e-01],
       [-1.30915756e+00,  1.17553721e-02,  2.18995346e-01,
         8.32347688e-01],
       [-1.37795313e+00,  2.85521442e-01,  3.38349190e-01,
         5.17145605e-01],
       [-2.38770162e+00,  2.45401324e-01,  3.44394580e-01,
         8.49736014e-01],
       [-9.13863820e-01, -7.16528760e-01, -6.98811614e-01,
        -1.03017182e-01],
       [ 1.36045196e-01,  3.33807238e-01,  4.72477727e-01,
         1.49135528e+00],
       [-6.39839709e-01, -8.87791348e-02,  4.37624163e-01,
         7.79246198e-01],
       [-1.84371659e+00,  6.80899548e-01,  8.06469153e-01,
         1.14071971e+00],
       [-1.41788513e+00,  3.63158778e-02,  7.95092158e-01,
         1.98274199e+00],
       [-1.73034527e+00, -1.63237781e+00, -2.29630346e-01,
         5.77824109e-01],
       [ 4.81646618e-01,  6.12349057e-01,  6.80612421e-01,
```

```
  1.55837454e+00],
[-1.00013075e+00, -1.10524493e-01,  1.63645136e-03,
  1.13056128e+00],
[-2.00655629e+00, -1.60028427e+00, -1.08580002e+00,
 -3.12311742e-01],
[-4.00309851e-02,  1.51336067e-01,  3.22188372e-01,
  6.08564790e-01],
[-1.40227144e+00, -1.14719659e+00, -4.48652091e-01,
  1.97159993e+00],
[-9.02400761e-02,  6.42781020e-01,  6.53779342e-01,
  1.36373116e+00],
[-1.73266469e+00, -3.26545824e-01,  3.52912847e-01,
  4.91417491e-01],
[-2.76642070e-01,  4.13747390e-01,  7.02119806e-01,
  1.46642745e+00],
[-9.73061840e-01, -4.09247352e-01,  5.90039576e-02,
  1.00271548e-01],
[-6.58089995e-01, -3.47310550e-01,  4.03904009e-01,
  6.84626936e-01],
[-9.89631028e-01, -8.64309581e-01,  3.27663794e-03,
  1.10425008e+00],
[-1.77325853e+00, -7.10095593e-01,  7.54284893e-01,
  8.49457161e-01],
[-4.75139877e-01, -4.07589701e-01, -1.26490375e-01,
  2.60124496e-02],
[-6.82296356e-01, -3.59227490e-01, -1.69561118e-01,
  4.80649154e-01],
[ 4.12277037e-01,  1.11446811e+00,  1.33764043e+00,
  1.61176354e+00],
[-4.32191613e-01, -2.66431737e-01, -6.60946938e-02,
  1.96387467e+00],
[-3.43937591e-01,  1.85274702e-01,  4.09633608e-01,
  1.59307658e+00],
[-2.41550572e+00, -7.36460145e-01, -5.54299889e-01,
  4.82435248e-01],
[-5.19079235e-01, -3.99733225e-01,  5.04197465e-01,
  6.82257807e-01],
[ 1.46030434e-01,  4.31623351e-01,  4.93509142e-01,
  1.04350533e+00],
[-1.15797870e+00, -4.08704073e-01,  3.31023845e-01,
  5.25844613e-01],
[-1.22575900e+00, -1.13708177e+00, -2.68434200e-01,
  7.06032003e-01],
```

```
[-4.82974603e-01,  4.69638023e-01,  4.76985566e-01,
  8.54545234e-01],
[-8.23698108e-01,  1.08396813e-01,  1.06580092e+00,
  2.01120216e+00],
[-7.29662037e-01, -4.56412641e-01, -1.83871773e-01,
  1.62694699e-01],
[-9.11003738e-01, -7.89848103e-01, -3.88862457e-01,
  4.83560789e-01],
[-2.10799716e+00, -4.09494148e-01, -3.08333953e-01,
 -2.83038681e-01],
[-9.33247754e-01,  6.17263344e-01,  1.80962109e+00,
  2.19012337e+00],
[-7.84737417e-01, -2.44042073e-01,  2.55177932e-01,
  3.42825107e-01],
[-1.62650171e+00, -3.00745070e-01, -2.85292452e-01,
  2.51078668e-01],
[-5.65573748e-01, -3.60080629e-01, -2.48041713e-01,
 -6.32709052e-02],
[-1.06014756e+00, -5.58112740e-02,  2.51651451e-01,
  6.75536498e-01],
[-7.51044211e-01,  1.86108116e-01,  4.88884239e-01,
  9.05559938e-01],
[-2.55864401e+00, -8.33907013e-01,  7.15775416e-01,
  2.23838618e+00],
[-2.35721931e+00, -1.28125964e+00, -7.15135564e-01,
 -6.88105713e-01],
[-1.05325411e+00, -4.19789678e-01,  9.77231895e-02,
  1.30859106e-01],
[-8.22547787e-01, -3.55008434e-01, -3.03939940e-01,
 -2.88374101e-01],
[-7.67208928e-01, -6.59537583e-01,  2.78764860e-01,
  6.98116466e-01],
[-2.29343223e+00, -6.84298754e-01, -2.90200289e-01,
  1.27151304e-01],
[-1.37800774e+00, -9.80812754e-01, -4.85001807e-01,
  1.02634614e+00],
[-3.86170952e-01, -3.12437928e-01, -7.93635000e-03,
  1.05984427e-01],
[-1.21735331e+00, -5.08619607e-01,  7.76632346e-02,
  1.55609784e+00],
[-7.67384993e-01, -4.24839010e-01,  1.96649387e-01,
  1.70530835e+00],
[ 2.67985474e-02,  1.69360237e-01,  2.25871477e-01,
```

```
  1.06995958e+00],
[-2.15094152e+00, -1.21126059e+00, -3.13831051e-01,
  5.52779988e-01],
[-1.53236671e+00, -8.44162265e-01, -7.23272511e-01,
 -1.46996022e-01],
[-2.53886811e-01,  2.58974499e-01,  6.79870256e-01,
  7.16933023e-01],
[-1.79094397e+00, -1.15981763e-01,  1.05018495e-01,
  1.60177718e+00],
[-5.06548923e-01, -4.15153421e-01, -3.61665257e-01,
 -2.98384656e-01],
[-1.49000279e+00, -1.07579922e+00,  1.74639209e-01,
  8.69213201e-01],
[-4.93281526e-01, -2.02375126e-01,  3.79079402e-02,
  1.15085141e-01],
[ 1.36566026e-02,  3.70733781e-02,  5.01195431e-01,
  1.23879342e+00],
[-1.89721089e+00, -6.10443771e-01, -3.91943134e-01,
  1.81315434e+00],
[-6.44654216e-01, -6.19238915e-01,  6.15312962e-01,
  1.20587875e+00],
[-6.57744835e-01, -1.04151009e-01,  6.08817647e-01,
  8.72202521e-01],
[-8.62785732e-01, -7.85905211e-01, -4.66196059e-01,
  3.76636712e-01],
[-2.14572410e+00, -1.33448899e+00,  5.65246548e-01,
  6.38624618e-01],
[-6.30745542e-01,  4.23867357e-01,  8.93278618e-01,
  1.26352546e+00],
[-1.58972643e+00, -8.19528455e-01,  6.66726684e-01,
  9.93841379e-01],
[-7.84301148e-01, -6.26358410e-01, -4.85491173e-01,
 -8.01123856e-02],
[-2.64547299e+00, -3.38626137e-01,  7.29100601e-02,
  1.35052662e+00],
[-1.29987756e+00, -3.00598605e-01, -2.55825750e-01,
  3.18725177e-01],
[-1.54770147e-01,  8.46134554e-02,  3.15352477e-01,
  7.84624644e-01],
[-2.18528723e+00, -4.78493653e-01,  4.04363441e-01,
  9.50306792e-01],
[-7.31170933e-01, -6.85441792e-01, -1.71497359e-01,
  1.34661105e+00],
```

```
[-2.20659291e+00, -2.82033471e-01,  5.56704006e-01,
  1.67187160e+00],
[-1.40535219e+00, -5.38640649e-01, -3.54224088e-01,
 -3.26459131e-01],
[-1.66014764e+00, -3.02177341e-01,  1.69553208e-01,
  1.36370184e+00],
[-1.27112709e+00, -7.07081008e-01, -6.30030109e-01,
 -3.72345168e-01],
[-1.69003273e+00, -1.21046313e+00, -6.82508051e-01,
 -1.09904622e-01],
[-1.28962940e+00, -1.00041576e+00,  1.13888968e+00,
  1.15981864e+00],
[-1.72082968e+00, -1.37816288e+00, -1.17858207e+00,
 -2.08758422e-01],
[-1.82861104e+00, -1.38478080e+00, -5.36981356e-02,
  1.99891132e+00],
[-1.13986819e+00,  1.34108358e-01,  1.60329036e-01,
  2.44716633e-01],
[-8.74156226e-01,  4.60400465e-03,  2.09190830e-01,
  8.10939629e-01],
[-1.46879138e+00,  1.01715095e+00,  1.05216972e+00,
  1.12128040e+00],
[-3.55286805e-01, -2.81297112e-01,  7.00960321e-02,
  3.22910049e-01],
[-9.69673438e-01, -7.73956341e-01, -1.41294362e-01,
  1.86497641e-01],
[-6.98252756e-01, -4.60867740e-01, -4.49205975e-01,
 -3.09809272e-01],
[-1.49856842e+00, -1.14555162e+00, -1.11438293e+00,
  6.21182151e-01],
[-1.83254993e+00, -1.35801881e+00,  4.28290437e-01,
  1.92338091e+00],
[-7.96594005e-01,  4.50047245e-01,  5.89911577e-01,
  1.21920578e+00],
[-9.00958281e-03,  3.82384161e-01,  5.04328532e-01,
  6.62491774e-01],
[-7.54788430e-02,  3.69944401e-01,  3.94644237e-01,
  1.39715609e+00],
[-1.03521162e+00,  1.99798306e-01,  2.22631076e-01,
  5.84166193e-01],
[-1.00661493e+00, -9.11311573e-01, -6.61437519e-01,
  2.63179766e-01],
[-6.52506034e-01,  1.64193815e-02,  3.53680451e-01,
```

```
        4.75966903e-01],
       [-5.42467799e-01,  1.29816242e-01,  1.40182556e-01,
        2.12232823e+00],
       [-8.11393093e-01,  3.28356583e-01,  5.48490558e-01,
        9.20939871e-01],
       [-7.72912147e-01, -6.92843228e-01, -4.83575763e-01,
        8.81266844e-03]], dtype=float64)
```

.at[]

Since the sampled object wraps a JAX array, you can use JAX's `.at[]` syntax for in-place style updates (which return a new object):

```
s = m.dist.normal(0,1, name = 'a', shape=(100,4, 5), sample = True, to_jax=False)
s_new = s.at[0, 0, 0].set(0.0)
```

reshape

```
m.dist.normal(0,1, name = 'a', shape=(100,4, 5), sample = True, to_jax=False).reshape((100, 1
```

```
SampledData([[[-2.11150516, -1.90744107, -0.32859384, ..., -0.08395863,
         0.91818652,  0.02864589],
       [ 1.07133933, -1.64409656,  0.11348187, ..., -2.23954389,
         0.31593896, -0.75168687],
       [ 0.13369154, -0.31710562, -1.21152002, ...,  0.42590341,
         0.76507914,  0.61751043],
       ...,
       [ 0.0492923 ,  0.48658905, -0.26981343, ...,  1.46055086,
         0.63734419,  1.01585025],
       [ 0.04668998,  1.71720245, -0.45420609, ..., -0.29365694,
         0.91761253, -0.80879681],
       [-0.06121082, -1.3455974 ,  0.18036684, ..., -0.62963005,
         0.19950374,  0.41596466]]], dtype=float64)
```

And many other standard NumPy/JAX array methods like `.flatten()`, `.squeeze()`, `.astype()`, etc., are also available directly on the sampled object.

Notes

> **Note**
>
> - `density`, `surface_3d`, and `ppc_plot` are **Plotly-only** — passing `interactive=False` has no effect on them.
> - The dimensional requirement is **enforced**: e.g. `corr_heatmap`, `boxplot`, `violinplot`, `pairplot`, `traceplot`, `scatter3d`, and `timeseries` raise a `ValueError` if the array is not the expected shape.
> - Plotting methods live on the `SampledData` wrapper only. If you sampled with `to_jax=True` (or later called `.to_jax()`), convert back with `SampledData(arr)` to plot, or re-sample without `to_jax`.