Distributions

Sampled from distributions

`Utils.np_dists.UnifiedDist` is a class to unify various distribution methods and provide a consistent interface for sampling and inference.

Asymmetric Laplace

Samples from an Asymmetric Laplace distribution.

The Asymmetric Laplace distribution is a generalization of the Laplace distribution, where the two sides of the distribution are scaled differently. It is defined by a location parameter (loc), a scale parameter (scale), and an asymmetry parameter (asymmetry).

$$f(x) = \frac{\text{asymmetry}}{\text{scale}(\text{asymmetry}^2+1)} \exp\left(-\frac{\text{asymmetry}}{\text{scale}}(\text{loc}-x)\right) \text{ if } x < \text{loc} \frac{\text{asymmetry}}{\text{scale}(\text{asymmetry}^2+1)} \exp\left(-\frac{1}{\text{scale}\cdot \text{asym}}\right.$$

Args:

```
bi.dist.asymmetric_laplace(
loc=0.0,
scale=1.0,
asymmetry=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
```

```
event=0,
create_obj=False,
)
```

- *loc* (jnp.ndarray or float): Location parameter of the distribution.
- *scale* (jnp.ndarray or float): Scale parameter of the distribution.
- *asymmetry* (jnp.ndarray or float): Asymmetry parameter of the distribution.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.
- validate_args (bool, optional): Whether to enable validation of distribution parameters. Defaults to `None`.

Returns:

- When `sample=False`: A BI AsymmetricLaplace distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BetaBinomial distribution (for direct sampling).
- When `create_obj=True`: The raw BI AsymmetricLaplace distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.asymmetric_laplace(loc=0.0, scale=1.0, asymmetry=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#asymmetriclaplace

---

Asymmetric Laplace Quantile

Samples from an AsymmetricLaplaceQuantile distribution.

This distribution is an alternative parameterization of the AsymmetricLaplace distribution, commonly used in Bayesian quantile regression. It utilizes a `quantile` parameter to define the balance between the left- and right-hand sides of the distribution, representing the proportion of probability density that falls to the left-hand side.

$$f(x) = \frac{1}{2\sigma} \exp\left(-\frac{|x-\mu|}{\sigma}\frac{1}{q-1}\right)\left(1-\frac{1}{2q}\right)$$

Args:

```
bi.dist.asymmetric_laplace_quantile(
loc=0.0,
scale=1.0,
quantile=0.5,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *loc* (float): The location parameter of the distribution.
- *sample* (float): The scale parameter of the distribution.

quantile (float): The quantile parameter, representing the proportion of probability density to the left of the median. Must be between 0 and 1.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI AsymmetricLaplaceQuantile distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BetaBinomial distribution (for direct sampling).
- When `create_obj=True`: The raw BI AsymmetricLaplaceQuantile distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.asymmetric_laplace_quantile(loc=0.0, scale=1.0, quantile=0.5, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#asymmetriclaplacequantile

---

Bernoulli

The Bernoulli distribution models a single trial with two possible outcomes: success or failure. It is parameterized by the probability of success, often denoted as 'p'.

$$P(X = 1) = pP(X = 0) = 1 - p$$

Args:

```
bi.dist.bernoulli(
probs=None,
logits=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *probs* (jnp.ndarray, optional): Probability of success for each Bernoulli trial. Must be between 0 and 1. logits (jnp.ndarray, optional): Log-odds of success for each Bernoulli trial. `probs = sigmoid(logits)`.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool, optional): Optional boolean array to mask observations.

- *create_obj* (bool, optional): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI Bernoulli distribution object (for model building) when `sample=False`. JAX array of samples drawn from the Bernoulli distribution (for direct sampling) when `sample=True`. The raw BI distribution object (for advanced use cases) when `create_obj=True`.

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.bernoulli(probs=0.7, sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#bernoulli

---

Bernoulli Logits

Samples from a Bernoulli distribution parameterized by logits.

The Bernoulli distribution models a single binary event (success or failure), parameterized by the log-odds ratio of success. The probability of success is given by the sigmoid function applied to the logit.

$$P(x) = \sigma(logits)$$

Args:

```
bi.dist.bernoulli_logits(
logits=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *logits* (jnp.ndarray, optional): Log-odds ratio of success. Must be real-valued.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int, optional): The number of batch dimensions to reinterpret as
- *event* dimensions (used in model building). Defaults to 0.
- *mask* (jnp.ndarray, bool, optional): Optional boolean array to mask observations. Defaults to None.
- *create_obj* (bool, optional): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`. Defaults to False.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI BernoulliLogits distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.bernoulli_logits(logits=jnp.array([0.2, 1, 2]), sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#bernoulli-logits

---

Bernoulli Probs

Samples from a Bernoulli distribution parameterized by probabilities.

The Bernoulli distribution models the probability of success in a single trial, where the outcome is binary (success or failure). It is characterized by a single parameter, `probs`, representing the probability of success.

$$P(X = 1) = p$$

where: p is the probability of success (0 <= p <= 1)

Args:

```
bi.dist.bernoulli_probs(
probs,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *probs* (jnp.ndarray): The probability of success for each Bernoulli trial. Must be between 0 and 1.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI BernoulliProbs distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).

- When `create_obj=True`: The raw BI BernoulliProbs object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.bernoulli_probs(probs=jnp.array([0.2, 0.7, 0.5]), sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#bernoulliprobs

---

Beta

Samples from a Beta distribution, defined on the interval [0, 1]. The Beta distribution is a versatile distribution often used to model probabilities or proportions. It is parameterized by two positive shape parameters, often referred to as concentration parameters in the BI context.

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$$

where α and β are the concentration parameters, and $B(x, y)$ is the Beta function.

Args:

```
bi.dist.beta(
concentration1,
concentration0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *concentration1* (jnp.ndarray): The first concentration parameter (shape parameter). Must be positive.
- *concentration0* (jnp.ndarray): The second concentration parameter (shape parameter). Must be positive.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Beta distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI Beta object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.beta(concentration1=1.0, concentration0=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#beta

BetaBinomial

Samples from a BetaBinomial distribution, a compound distribution where the probability of success in a binomial experiment is drawn from a Beta distribution. This models situations where the underlying probability of success is not fixed but varies according to a prior belief represented by the Beta distribution.

$$P(X = k) = \binom{n}{k} \frac{\Gamma(\alpha + k)}{\Gamma(\alpha + \beta + n - k)} \frac{\Gamma(\beta + n - k)}{\Gamma(\beta)}$$

Args:

```
bi.dist.beta_binomial(
concentration1,
concentration0,
total_count=1,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *concentration1* (jnp.ndarray): The first concentration parameter (alpha) of the Beta distribution.
- *concentration0* (jnp.ndarray): The second concentration parameter (beta) of the Beta distribution.
- *total_count* (jnp.ndarray): The number of Bernoulli trials in the Binomial part of the distribution.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI BetaBinomial distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BetaBinomial distribution (for direct sampling).
- When `create_obj=True`: The raw BI BetaBinomial distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.beta_binomial(concentration1=1.0, concentration0=1.0, total_count=10, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#betabinomial

---

Beta Proportion

The BetaProportion distribution is a reparameterization of the conventional Beta distribution in terms of a the variate mean and a precision parameter. It's useful for modeling rates and proportions.

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$$

Args:

```
bi.dist.beta_proportion(
mean,
concentration,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *mean* (jnp.ndarray): The mean of the BetaProportion distribution,must be between 0 and 1.
- *concentration* (jnp.ndarray): The concentration parameter of the BetaProportion distribution.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI BetaProportion distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BetaBinomial distribution (for direct sampling).
- When `create_obj=True`: The raw BI BetaProportion distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
samples = m.dist.beta_proportion(mean=0.5, concentration=2.0, sample=True, shape=(1000,))
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#beta_proportion

---

Binomial

The Binomial distribution models the number of successes in a sequence of independent Bernoulli trials. It represents the probability of obtaining exactly k successes in n trials, where each trial has a probability p of success.

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Args:

```
bi.dist.binomial(
total_count=1,
probs=None,
logits=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *total_count* (int): The number of trials n.
- *probs* (jnp.ndarray, optional): The probability of success p for each trial. Must be between 0 and 1.
- *logits* (jnp.ndarray, optional): The log-odds of success for each trial. `probs = jax.nn.sigmoid(logits)`.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool, optional): Optional boolean array to mask observations.
- *create_obj* (bool, optional): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.

- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

Binomial distribution object (for model building) when `sample=False`. JAX array of samples drawn from the Binomial distribution (for direct sampling) when `sample=True`. The raw BI distribution object (for advanced use cases) when `create_obj=True`.

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.binomial(total_count=10, probs=0.5, sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#binomial

---

Binomial Logits

The BinomialLogits distribution represents a binomial distribution parameterized by logits. It is useful when the probability of success is not directly known but is instead expressed as logits, which are the natural logarithm of the odds ratio.

$$P(X = k) = \binom{n}{k} \frac{e^{logits_k}}{1 + e^{logits_k}}$$

Args:

```
bi.dist.binomial_logits(
logits,
total_count=1,
validate_args=None,
name='x',
```

```
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

logits (jnp.ndarray): Log-odds of each success. total_count (int): Number of trials.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI BinomialLogits distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI BinomialLogits object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.binomial_logits(logits=jnp.zeros(10), total_count=5, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#binomialllogits

---

Binomial Probs

Samples from a Binomial distribution with specified probabilities for each trial.

The Binomial distribution models the number of successes in a sequence of independent Bernoulli trials, where each trial has the same probability of success.

$$P(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Args:

```
bi.dist.binomial_probs(
probs,
total_count=1,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *probs* (jnp.ndarray): The probability of success for each trial. Must be between 0 and 1.
- *total_count* (int): The number of trials in each sequence.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI BinomialProbs distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI BinomialLogits object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.binomial_probs(probs=0.5, total_count=10, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#binomialprobs

---

Conditional Autoregressive (CAR)

The CAR distribution models a vector of variables where each variable is a linear combination of its neighbors in a graph.

$$p(x) = \prod_{i=1}^{K} \mathcal{N}(x_i|\mu_i, \Sigma_i)$$

where μ_i is a function of the values of the neighbors of site i and Σ_i is the variance of site i.

.. note::

The CAR distribution is a special case of the multivariate normal distribution. It is used to model spatial data, such as temperature or precipitation.

Args:

```
bi.dist.car(
loc,
correlation,
conditional_precision,
adj_matrix,
is_sparse=False,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *loc* (Union[float, Array]): Mean of the distribution.
- *correlation* (Union[float, Array]): Correlation between variables.
- *conditional_precision* (Union[float, Array]): Precision of the distribution.
- *adj_matrix* (Union[Array, scipy.sparse.spmatrix]): Adjacency matrix defining the graph. is_sparse (bool): Whether the adjacency matrix is sparse. Defaults to False.
- *validate_args/ (bool): Whether to validate arguments. Defaults to None.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI CAR distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI CAR object (for advanced use cases).

Categorical distribution.

The Categorical distribution, also known as the multinomial distribution, describes the probability of different outcomes from a finite set of possibilities. It is commonly used to model discrete choices or classifications.

$$P(k) = \frac{e^{\log(p_k)}}{\sum_{j=1}^{K} e^{\log(p_j)}}$$

where p_k is the probability of outcome k, and the sum is over all possible outcomes.

Args:

```
bi.dist.categorical(
probs=None,
logits=None,
validate_args=None,
name='x',
obs=None,
mask=None,
```

```
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *probs* (jnp.ndarray): A 1D array of probabilities for each category. Must sum to 1.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Categorical distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the Categorical distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.categorical(probs=jnp.array([0.2, 0.3, 0.5]), sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#categorical

---

Categorical Logits

Samples from a Categorical distribution with logits. This distribution represents a discrete probability distribution over a finite set of outcomes, where the probabilities are determined by the logits. The probability of each outcome is given by the softmax function applied to the logits.

$$P(k) = \frac{e^{logits_k}}{\sum_{j=1}^{K} e^{logits_j}}$$

Args:

```
bi.dist.categorical_logits(
logits,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *logits* (jnp.ndarray): Log-odds of each category.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.

- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI CategoricalLogits distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI CategoricalLogits object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.categorical_logits(logits=jnp.zeros(5), sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#categoricallogits

---

Categorical Probs distribution.

Samples from a Categorical distribution.

The Categorical distribution is a discrete probability distribution that represents the probability of each outcome from a finite set of possibilities. It is often used to model the outcome of a random process with a fixed number of possible outcomes, such as the roll of a die or the selection of an item from a list.

$$P(x) = \frac{probs_i}{\sum_{k=1}^{K} probs_k}$$

Args:

```
bi.dist.categorical_probs(
probs,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *probs* (jnp.ndarray): Probabilities for each category. Must sum to 1.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.

- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI CategoricalProbs distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI CategoricalProbs object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.categorical_probs(probs=jnp.array([0.2, 0.3, 0.5]), sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#categoricalprobs

---

Cauchy

Samples from a Cauchy

The Cauchy distribution, also known as the Lorentz distribution, is a continuous probability distribution that arises frequently in various fields, including physics and statistics. It is characterized by its heavy tails, which extend indefinitely.

$$f(x) = \frac{1}{\pi\gamma}\left[\frac{\gamma^2}{(x-\mu)^2+\gamma^2}\right]$$

Args:

```
bi.dist.cauchy(
loc=0.0,
scale=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *loc* (jnp.ndarray or float, optional): Location parameter. Defaults to 0.0.
- *sample* (jnp.ndarray or float, optional): Scale parameter. Must be positive. Defaults to 1.0.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int, optional): The number of batch dimensions to reinterpret as event dimensions (used in model building). Defaults to None.
- *mask* (jnp.ndarray, bool, optional): Optional boolean array to mask observations. Defaults to None.
- *create_obj* (bool, optional): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`. Defaults to False.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Cauchy distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the Cauchy distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.cauchy(loc=0.0, scale=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#cauchy

Chi-squared

The Chi-squared distribution is a continuous probability distribution that arises frequently in hypothesis testing, particularly in ANOVA and chi-squared tests. It is defined by a single positive parameter, degrees of freedom (df), which determines the shape of the distribution.

$$p(x; df) = \frac{1}{2^{df/2}\Gamma(df/2)} x^{df/2-1} e^{-x/2}$$

Args:

```
bi.dist.chi2(
df,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
```

```
event=0,
create_obj=False,
)
```

df (jnp.ndarray): Degrees of freedom. Must be positive.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Chi2 distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI Chi2 object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu)
m.dist.chi2(df=3.0, sample = True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#chi2

Circulant Normal Multivariate normal

Circulant Normal Multivariate normal distribution with covariance matrix $\mathbf{C}$ that is positive-definite and circulant [1], i.e., has periodic boundary conditions. The density of a sample $\mathbf{x} \in \mathbb{R}^n$ is the standard multivariate normal density

$$p\left(\mathbf{x} \mid \mu, \mathbf{C}\right) = \frac{\left(\det \mathbf{C}\right)^{-1/2}}{\left(2\pi\right)^{n/2}} \exp\left(-\frac{1}{2}\left(\mathbf{x}-\mu\right)^{\top} \mathbf{C}^{-1}\left(\mathbf{x}-\mu\right)\right),$$

where det denotes the determinant and $^{\top}$ the transpose. Circulant matrices can be diagnolized efficiently using the discrete Fourier transform [1], allowing the log likelihood to be evaluated in $n \log n$ time for n observations [2].

- *loc:* Mean of the distribution μ.
- *covariance_row:* First row of the circulant covariance matrix C. Because of periodic boundary conditions, the covariance matrix is fully determined by its first row (see :func:`jax.scipy.linalg.toeplitz` for further details).
- *covariance_rfft:* Real part of the real fast Fourier transform of :code:`covariance_row`, the first row of the circulant covariance matrix C.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

References:

1. Wikipedia. (n.d.). Circulant matrix. Retrieved March 6, 2025, from https://en.wikipedia.org/wiki/Circula
2. Wood, A. T. A., & Chan, G. (1994). Simulation of Stationary Gaussian Processes in $[0, 1]^d$. *Journal of Computational and Graphical Statistics*, 3(4), 409–432. https://doi.org/10.1080/10618600.1994.10474655

Args:

```
bi.dist.circulant_normal(
loc: jax.Array,
covariance_row: jax.Array = None,
covariance_rfft: jax.Array = None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *loc* : jnp.ndarray Mean of the distribution μ.
- *covariance_row* : jnp.ndarray, optional. First row of the circulant covariance matrix $\mathbf{C}$. Defaults to None.
- *covariance_rfft* : jnp.ndarray, optional Real part of the real fast Fourier transform of :code:`covariance_row`. Defaults to None.

Returns:

- When `sample=False`: A BI Circulant Normal Distribution distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI Circulant Normal Distribution object (for advanced use cases).

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#normal

---

Delta

The Delta distribution, also known as a point mass distribution, assigns probability 1 to a single point and 0 elsewhere. It's useful for representing deterministic variables or as a building block for more complex distributions.

$$P(x = v) = 1$$

Args:

```
bi.dist.delta(
v=0.0,
log_density=0.0,
event_dim=0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

v (jnp.ndarray): The location of the point mass. log_density (float, optional): The log probability density of the point mass. This is primarily for creating distributions that are non-normalized or for specific advanced use cases. For a standard delta distribution, this should be 0. Defaults to 0.0.
event_dim (int, optional): The number of rightmost dimensions of `v` to interpret as event dimensions. Defaults to 0. - *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.

- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Circulant Delta Distribution distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI Circulant Delta Distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.delta(v=0.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#delta

Dirichlet

Samples from a Dirichlet distribution.

The Dirichlet distribution is a multivariate generalization of the Beta distribution. It is a probability distribution over a simplex, which is a set of vectors where each element is non-negative and sums to one. It is often used as a prior distribution for categorical distributions.

$$P(x_1, ..., x_K) = \frac{\Gamma(\sum_{i=1}^{K} \alpha_i)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}$$

Args:

```
bi.dist.dirichlet(
concentration,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

concentration (jnp.ndarray): The concentration parameter(s) of the Dirichlet distribution. Must be a positive array.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Dirichlet Distribution distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI Dirichlet Distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.dirichlet(concentration=jnp.array([1.0, 1.0, 1.0]), sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#dirichlet

---

Dirichlet-Multinomial

Creates a Dirichlet-Multinomial compound distribution, which is a Multinomial distribution with a Dirichlet prior on its probabilities. It is often used in Bayesian statistics to model count data where the proportions of categories are uncertain.

The probability mass function is given by:

$$P(\mathbf{x}|\alpha, n) = \frac{n!}{\prod_{i=1}^{k} x_i!} \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\Gamma(n + \sum_{i=1}^{k} \alpha_i)} \prod_{i=1}^{k} \frac{\Gamma(x_i + \alpha_i)}{\Gamma(\alpha_i)}$$

where $\mathbf{x}$ is a vector of counts, n is the total number of trials (`total_count`), and α is the `concentration` parameter vector for the Dirichlet prior.

Args:

```
bi.dist.dirichlet_multinomial(
concentration,
total_count=1,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

concentration (jnp.ndarray): The concentration parameter (alpha) for the Dirichlet prior. Values must be positive. The last dimension is interpreted as the number of categories. total_count (int, jnp.ndarray, optional): The total number of trials (n). This must be a non-negative integer. Defaults to 1. validate_args (bool, optional): Whether to enable validation of distribution parameters. Defaults to `None`. - *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'. - *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.

- *mask* (jnp.ndarray, bool, optional): Optional boolean array to mask observations. If provided, events with a `True` mask will be conditioned on `obs`, while the remaining events will be treated as latent variables. Defaults to `None`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. This argument has no effect when `sample=False`. Defaults to 0.
- *shape* (tuple, optional): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int, optional): The number of batch dimensions to reinterpret as event dimensions (used in model building).

- *create_obj* (bool, optional): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`. Defaults to `False`.

Returns:

- When `sample=False`: A BI dirichlet_multinomial Distribution distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI dirichlet_multinomial Distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
import jax.numpy as jnp
m = bi('cpu')

# Direct sampling
# Sample a single vector of counts for 10 trials from 3 categories
counts = m.dist.dirichlet_multinomial(concentration=jnp.array([1.0, 1.0, 1.0]),total_count=1

# Usage within a model
def my_model(obs_data=None):
# Define a prior on the concentration parameter
alpha = m.dist.half_cauchy(scale=jnp.ones(5), name='alpha', shape=(5,))

# Model observed counts
with m.plate('data', len(obs_data)):
y = m.dist.dirichlet_multinomial(
concentration=alpha,
total_count=100,
name='y',
obs=obs_data
)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#dirichletmultinomial

---

Discrete Uniform

Samples from a Discrete Uniform distribution.

The Discrete Uniform distribution defines a uniform distribution over a range of integers. It is characterized by a lower bound (`low`) and an upper bound (`high`), inclusive.

$$P(X = k) = \frac{1}{high - low + 1}, \quad k \in \{low, low + 1, ..., high\}$$

Args:

```
bi.dist.discrete_uniform(
low=0,
high=1,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

low (jnp.ndarray): The lower bound of the uniform range, inclusive. high (jnp.ndarray): The upper bound of the uniform range, inclusive.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI DiscreteUniform Distribution distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI dirichlet_multinomial Distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.discrete_uniform(low=0, high=5, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#discreteuniform

---

Doubly Truncated Power Law

This distribution represents a continuous power law with a finite support bounded between `low` and `high`, and with an exponent `alpha`. It is normalized over the interval `[low, high]` to ensure the area under the density function is 1.

The probability density function (PDF) is defined as:

$$f(x;\,\alpha,a,b)=\frac{x^{\alpha}}{Z(\alpha,a,b)},\quad x\in[a,b]$$

where the normalization constant $Z(\alpha,a,b)$ is given by

$$Z(\alpha,a,b)=\begin{cases}\log(b)-\log(a), & \text{if } \alpha=-1,\\ \dfrac{b^{1+\alpha}-a^{1+\alpha}}{1+\alpha}, & \text{otherwise.}\end{cases}$$

This distribution is useful for modeling data that follows a power-law behavior but is naturally bounded due to measurement or theoretical constraints (e.g., finite-size systems).

Args:

```
bi.dist.doubly_truncated_power_law(
alpha,
low,
high,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

alpha (float or array-like): Power-law exponent.

low (float or array-like): Lower bound of the distribution (must be ≥ 0).

high (float or array-like): Upper bound of the distribution (must be > 0).

- *shape* (tuple, optional): The shape of the output tensor. Defaults to None.

validate_args (bool, optional): Whether to validate the arguments. Defaults to True.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.

- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI doubly_truncated_power_law Distribution distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the BernoulliLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI doubly_truncated_power_law Distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.doubly_truncated_power_law(low=0.1, high=10.0, alpha=2.0, sample=True)
```

Euler–Maruyama

Euler–Maruyama methode is a method for the approximate numerical solution of a stochastic differential equation (SDE). It simulates the solution to an SDE by iteratively applying the Euler method to each time step, incorporating a random perturbation to account for the diffusion term.

$$dX_t = f(X_t, t)dt + g(X_t, t)dW_t$$

where: - X_t is the state of the system at time t. - $f(X_t, t)$ is the drift coefficient. - $g(X_t, t)$ is the diffusion coefficient. - dW_t is a Wiener process (Brownian motion).

Args:

```
bi.dist.euler_maruyama(
t,
sde_fn,
init_dist,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

t (jnp.ndarray): Discretized time steps.

sde_fn (callable): A function that takes the current state and time as input and returns the drift and diffusion coefficients. init_dist (Distribution): The initial distribution of the system.

- *shape* (tuple, optional): The shape of the output tensor. Defaults to None.

sample_shape (tuple, optional): The shape of the samples to draw. Defaults to None.

validate_args (bool, optional): Whether to validate the arguments. Defaults to True.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

jnp.ndarray: Samples drawn from the Euler–Maruyama distribution.

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.euler_maruyama(t=jnp.array([0.0, 0.1, 0.2]), sde_fn=lambda x, t: (x, 1.0), init_dist=
```

Exponential

The Exponential distribution is a continuous probability distribution that models the time until an event occurs in a Poisson process, where events occur continuously and independently at a constant average rate. It is often used to model the duration of events, such as the time until a machine fails or the length of a phone call.

$$f(x) = \lambda e^{-\lambda x} \text{ for } x \geq 0$$

Args:

```
bi.dist.exponential(
rate=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

rate (jnp.ndarray): The rate parameter, λ. Must be positive.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).

- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Exponential distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the Exponential distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.exponential(rate=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#exponential

Folded

Samples from a Folded distribution, which is the absolute value of a base univariate distribution. This distribution reflects the base distribution across the origin, effectively taking the absolute value of each sample.

$$p(x) = \sum_{k=-\infty}^{\infty} p(x - 2k)$$

Args:

```
bi.dist.folded_distribution(
base_dist,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *loc* (float, optional): Location parameter of the base distribution. Defaults to 0.0.
- *sample* (float, optional): Scale parameter of the base distribution. Defaults to 1.0.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI FoldedDistribution distribution object (for model building) when `sample=False`. JAX array of samples drawn from the FoldedDistribution distribution (for direct sampling) when `sample=True`. The raw BI distribution object (for advanced use cases) when `create_obj=True`.

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.folded_distribution(m.dist.normal(loc=0.0, scale=1.0, create_obj = True), sample=True
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#foldeddistribution

---

Gamma

Samples from a Gamma distribution.

The Gamma distribution is a continuous probability distribution that arises frequently in Bayesian statistics, particularly in prior distributions for variance parameters. It is defined by two positive shape parameters, concentration (k) and rate (theta).

$$f(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \quad x > 0$$

Args:

```
bi.dist.gamma(
concentration,
rate=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
```

```
shape=(),
event=0,
create_obj=False,
)
```

concentration (jnp.ndarray): The shape parameter of the Gamma distribution (k > 0).

rate (jnp.ndarray): The rate parameter of the Gamma distribution (theta > 0).

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

Gamma: A BI Gamma distribution object (for model building).

jnp.ndarray: A JAX array of samples drawn from the Gamma distribution (for direct sampling).

Gamma: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.gamma(concentration=2.0, rate=0.5, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#gamma

Gamma Poisson

A compound distribution comprising of a gamma-poisson pair, also referred to as a gamma-poisson mixture. The `rate` parameter for the :class:`~numpyro.distributions.Poisson` distribution is unknown and randomly drawn from a :class:`~numpyro.distributions.Gamma` distribution.

$$P(X = x) = \int_0^\infty \frac{1}{x} \exp(-x\lambda) \frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x\beta} dx$$

Args:

```
bi.dist.gamma_poisson(
concentration,
rate=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

concentration (jnp.ndarray): Shape parameter (alpha) of the Gamma distribution. rate (jnp.ndarray): Rate parameter (beta) for the Gamma distribution.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI GammaPoisson distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the GammaPoisson distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.gamma_poisson(concentration=1.0, rate=2.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#gammapoisson

---

Gaussian Copula

A distribution that links the `batch_shape[:-1]` of a marginal distribution with a multivariate Gaussian copula, modelling the correlation between the axes. A copula is a multivariate distribution over the uniform distribution on [0, 1]. The Gaussian copula links the marginal distributions through a multivariate normal distribution.

$$f(x_1, ..., x_d) = \prod_{i=1}^{d} f_i(x_i) \cdot \phi(F_1(x_1), ..., F_d(x_d); \mu, \Sigma)$$

where: - f_i is the probability density function of the i-th marginal distribution. - F_i is the cumulative distribution function of the i-th marginal distribution. - ϕ is the standard normal PDF. - μ is the mean vector of the multivariate normal distribution. - Σ is the covariance matrix of the multivariate normal distribution.

Args:

```
bi.dist.gaussian_copula(
marginal_dist,
correlation_matrix=None,
correlation_cholesky=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

marginal_dist (Distribution): Distribution whose last batch axis is to be coupled.

correlation_matrix (array_like, optional): Correlation matrix of the coupling multivariate normal distribution. Defaults to None.

correlation_cholesky (array_like, optional): Correlation Cholesky factor of the coupling multivariate normal distribution. Defaults to None.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool, optional): Optional boolean array to mask observations. Defaults to None.
- *create_obj* (bool, optional): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`. Defaults to False.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI GaussianCopula distribution object: When `sample=False` (for model building). JAX array: When `sample=True` (for direct sampling). BI distribution object: When `create_obj=True` (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.gaussian_copula(
marginal_dist = m.dist.beta(2.0, 5.0, create_obj = True),
correlation_matrix = jnp.array([[1.0, 0.7],[0.7, 1.0]]),
sample = True
)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#gaussiancopula

---

Gaussian Copula Beta

This distribution combines a Gaussian copula with a Beta distribution. The Gaussian copula models the dependence structure between random variables, while the Beta distribution defines the marginal distributions of each variable.

$$f(x) = \int_{-\infty}^{\infty} g(x|u)h(u)du$$

Where: - g(x|u) is the Gaussian copula density. - h(u) is the Beta density.

Args:

```
bi.dist.gaussian_copula_beta(
concentration1,
concentration0,
correlation_matrix=None,
correlation_cholesky=None,
validate_args=False,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

concentration1 (jnp.ndarray): The first shape parameter of the Beta distribution.

concentration0 (jnp.ndarray): The second shape parameter of the Beta distribution.

correlation_matrix (array_like, optional): Correlation matrix of the coupling multivariate normal distribution. Defaults to None.

correlation_cholesky (jnp.ndarray): The Cholesky decomposition of the correlation matrix. - *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.

- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

GaussianCopulaBeta: A BI GaussianCopulaBeta distribution object (for model building). jnp.ndarray: A JAX array of samples drawn from the GaussianCopulaBeta distribution (for direct sampling). Distribution: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.gaussian_copula_beta(
concentration1 = jnp.array([2.0, 3.0]),
concentration0 = jnp.array([5.0, 3.0]),
correlation_cholesky = jnp.linalg.cholesky(jnp.array([[1.0, 0.7],[0.7, 1.0]])),
sample = True
)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#gaussiancopulabetadistribution

---

Gaussian Random Walk

Creates a distribution over a Gaussian random walk of a specified number of steps. This is a discrete-time stochastic process where the value at each step is the previous value plus a Gaussian-distributed increment. The distribution is over the entire path.

$$X_t = X_{t-1} + \epsilon_t, \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2)$$

with the initial state $X_0 = 0$. The resulting sample is a vector of length `num_steps`, representing the path $(X_1, X_2, \ldots, X_{\text{num\_steps}})$.

Args:

```
bi.dist.gaussian_random_walk(
scale=1.0,
num_steps=1,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *sample* (float, jnp.ndarray, optional): The standard deviation (σ) of the Gaussian increments. Must be positive. Defaults to 1.0. num_steps (int, optional): The number of steps in the random walk, which determines the event shape of the distribution. Must be positive. Defaults to 1. validate_args (bool, optional): Whether to enable validation of distribution parameters. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *mask* (jnp.ndarray, bool, optional): Optional boolean array to mask observations. If provided, events with a `True` mask will be conditioned on `obs`, while the remaining events will be treated as latent variables. Defaults to `None`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. This argument has no effect when `sample=False`. Defaults to 0.
- *shape* (tuple, optional): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int, optional): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *create_obj* (bool, optional): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`. Defaults to `False`.

Returns:

BI.primitives.Messenger: A BI sample site object when used in a model context (`sample=False`). jnp.ndarray: A JAX array of samples drawn from the GaussianRandomWalk distribution (for direct sampling, `sample=True`). numpyro.distributions.Distribution: The raw BI distribution object (if `create_obj=True`).

Example Usage:

```
from BI import bi
import jax.numpy as jnp
m = bi('cpu')

# Direct sampling of a random walk with 100 steps
path = m.dist.gaussian_random_walk(scale=0.5, num_steps=100, sample=True)

# Usage within a model for a latent time series
def my_model(data=None):
# Prior on the volatility of the random walk
volatility = m.dist.half_cauchy(scale=1.0, name='volatility')
```

```
# The latent random walk
latent_process = m.dist.gaussian_random_walk(
scale=volatility,
num_steps=len(data) if data is not None else 10,
name='latent_process'
)

# Observation model
# Assumes the observed data is the latent process plus some noise
obs_noise = m.dist.half_cauchy(scale=1.0, name='obs_noise')
with m.plate('time', len(data) if data is not None else 10):
return m.dist.normal(loc=latent_process, scale=obs_noise, obs=data, name='obs')
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#gaussianrandomwalk

---

Gaussian State Space

Samples from a Gaussian state space model.

$$\mathbf{z}_t = \mathbf{A}\mathbf{z}_{t-1} + \epsilon_t = \sum_{k=1} \mathbf{A}^{t-k}\epsilon_t,$$

where $\mathbf{z}_t$ is the state vector at step t, $\mathbf{A}$ is the transition matrix, and ϵ is the innovation noise.

Args:

```
bi.dist.gaussian_state_space(
num_steps,
transition_matrix,
covariance_matrix=None,
precision_matrix=None,
scale_tril=None,
validate_args=None,
name='x',
obs=None,
```

```
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

num_steps (int): Number of steps.

transition_matrix (jnp.ndarray): State space transition matrix $\mathbf{A}$.

covariance_matrix (jnp.ndarray, optional): Covariance of the innovation noise ϵ. Defaults to None.

precision_matrix (jnp.ndarray, optional): Precision matrix of the innovation noise ϵ. Defaults to None.

scale_tril (jnp.ndarray, optional): Scale matrix of the innovation noise ϵ. Defaults to None.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool, optional): Optional boolean array to mask observations. Defaults to None.
- *create_obj* (bool, optional): If True, returns the raw BI distribution object instead of creating a sample site. Defaults to False.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI GaussianStateSpace distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the GaussianStateSpace distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.gaussian_state_space(num_steps=5, transition_matrix=jnp.array([[0.5]]), covariance_mat
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#gaussianstate

---

Geometric distribution.

The Geometric distribution models the number of failures before the first success in a sequence of Bernoulli trials. It is characterized by a single parameter, the probability of success on each trial.

$$P(X = k) = (1 - p)^k p$$

Args:

```
bi.dist.geometric(
probs=None,
logits=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
```

```
create_obj=False,
)
```

- *probs* (jnp.ndarray, optional): Probability of success on each trial. Must be between 0 and 1. logits (jnp.ndarray, optional): Log-odds of success on each trial. `probs = jax.nn.sigmoid(logits)`.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool, optional): Optional boolean array to mask observations.
- *create_obj* (bool, optional): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Geometric distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the Geometric distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.geometric(probs=0.5, sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#geometric

---

GeometricLogits

Samples from a GeometricLogits distribution, which models the number of failures before the first success in a sequence of independent Bernoulli trials. It is parameterized by logits, which are transformed into probabilities using the sigmoid function.

$$P(X = k) = (1 - p)^k p$$

where:

- X is the number of failures before the first success.
- k is the number of failures.
- p is the probability of success on each trial (derived from the logits).

Args:

```
bi.dist.geometric_logits(
logits,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

logits (jnp.ndarray): Log-odds parameterization of the probability of success.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI GeometricLogits distribution object (for model building) when `sample=False`. JAX array of samples drawn from the GeometricLogits distribution (for direct sampling) when `sample=True`. The raw BI distribution object (for advanced use cases) when `create_obj=True`.

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.geometric_logits(logits=jnp.zeros(10), sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#geometriclogits

---

GeometricProbs

Samples from a Geometric

The Geometric distribution models the number of trials until the first success in a sequence of independent Bernoulli trials, where each trial has the same probability of success.

$$P(X = k) = (1 - p)^k p$$

Args:

```
bi.dist.geometric_probs(
probs,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *probs* (jnp.ndarray): Probability of success on each trial. Must be between 0 and 1.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.

- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI GeometricProbs distribution object (for model building). JAX array of samples drawn from the GeometricProbs distribution (for direct sampling). The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.geometric_probs(probs=0.5, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#geometricprobs

---

Gompertz

The Gompertz distribution is a distribution with support on the positive real line that is closely related to the Gumbel distribution. This implementation follows the notation used in the Wikipedia entry for the Gompertz distribution. See https://en.wikipedia.org/wiki/Gompertz_distribution.

The probability density function (PDF) is:

$$f(x) = \frac{con}{rate} \exp\left\{-\frac{con}{rate}\left[\exp\{x * rate\} - 1\right]\right\} \exp(-x * rate)$$

Args:

```
bi.dist.gompertz(
concentration,
rate=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

concentration (jnp.ndarray): The concentration parameter. Must be positive. rate (jnp.ndarray): The rate parameter. Must be positive.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI Gompertz distribution object: When `sample=False` (for model building). JAX array: When `sample=True` (for direct sampling). BI distribution object: When `create_obj=True` (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.gompertz(concentration=1.0, rate=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#gompertz

Gumbel

Samples from a Gumbel (or Extreme Value) distribution.

The Gumbel distribution is a continuous probability distribution named after German mathematician Carl Gumbel. It is often used to model the distribution of maximum values in a sequence of independent random variables.

$$f(x) = \frac{1}{s} e^{-(x-\mu)/s} e^{-e^{-(x-\mu)/s}}$$

Args:

```
bi.dist.gumbel(
loc=0.0,
scale=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
```

```
event=0,
create_obj=False,
)
```

- *loc* (jnp.ndarray or float, optional): Location parameter. Defaults to 0.0.
- *sample* (jnp.ndarray or float, optional): Scale parameter. Must be positive. Defaults to 1.0.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int, optional): The number of batch dimensions to reinterpret as event dimensions (used in model building). Defaults to 1.
- *mask* (jnp.ndarray, bool, optional): Optional boolean array to mask observations. Defaults to None.
- *create_obj* (bool, optional): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`. Defaults to False.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Gumbel distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the Gumbel distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.gumbel(loc=0.0, scale=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#gumbel

HalfCauchy

The HalfCauchy distribution is a probability distribution that is half of the Cauchy distribution. It is defined on the positive real numbers and is often used in situations where only positive values are relevant.

$$f(x) = \frac{1}{2} \cdot \frac{1}{\pi \cdot \frac{1}{scale} \cdot (x^2 + \frac{1}{scale^2})}$$

Args:

```
bi.dist.half_cauchy(
scale=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *sample* (jnp.ndarray): The scale parameter of the Cauchy distribution. Must be positive.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.

- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI HalfCauchy distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the HalfCauchy distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.half_cauchy(scale=1.0, sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#halfcauchy

---

HalfNormal

Samples from a HalfNormal distribution.

The HalfNormal distribution is a distribution of the absolute value of a normal random variable. It is defined by a location parameter (implicitly 0) and a scale parameter.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \text{ for } x > 0$$

Args:

```
bi.dist.half_normal(
scale=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *sample* (float, array): The scale parameter of the distribution. Must be positive.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.

- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI HalfNormal distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the HalfNormal distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.half_normal(scale=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#halfnormal

Improper Uniform

A helper distribution with zero :meth:`log_prob` over the `support` domain.

$$p(x) = 0$$

Args:

```
bi.dist.improper_uniform(
support,
batch_shape,
event_shape,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

support (numpyro.distributions.constraints.Constraint): The support of this distribution.

batch_shape (tuple): Batch shape of this distribution. It is usually safe to set `batch_shape=()`.

event_shape (tuple): Event shape of this distribution.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI ImproperUniform distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the ImproperUniform distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import sample
from numpyro.distributions import ImproperUniform, Normal, constraints

def model():
x = sample('x', ImproperUniform(constraints.ordered_vector, (), event_shape=(10,)))
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#improperuniform

---

Inverse Gamma

The InverseGamma distribution is a two-parameter family of continuous probability distributions. It is defined by its shape and rate parameters. It is often used as a prior distribution for precision parameters (inverse variance) in Bayesian statistics.

$$p(x) = \frac{1}{Gamma(\alpha)} \left(\frac{\beta}{\Gamma(\alpha)} \right)^{\alpha} x^{\alpha-1} e^{-\beta x} \text{ for } x > 0$$

Args:

```
bi.dist.inverse_gamma(
concentration,
rate=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
```

```
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

concentration (jnp.ndarray): The shape parameter (\alpha) of the InverseGamma distribution. Must be positive. rate (jnp.ndarray): The rate parameter (\beta) of the InverseGamma distribution. Must be positive.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI InverseGamma distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the InverseGamma distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.inverse_gamma(concentration=2.0, rate=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#inversegamma

---

Kumaraswamy

The Kumaraswamy distribution is a continuous probability distribution defined on the interval [0, 1]. It is a flexible distribution that can take on various shapes depending on its parameters.

$$f(x; a, b) = abx^{ab-1}(1-x)^{b-1}$$

Args:

```
bi.dist.kumaraswamy(
concentration1,
concentration0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

concentration1 (jnp.ndarray): The first shape parameter. Must be positive. concentration0 (jnp.ndarray): The second shape parameter. Must be positive.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Kumaraswamy distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the Kumaraswamy distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.kumaraswamy(concentration1=2.0, concentration0=3.0, sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#kumaraswamy

---

Laplace

Samples from a Laplace distribution, also known as the double exponential distribution. The Laplace distribution is defined by its location parameter (loc) and scale parameter (scale).

$$f(x) = \frac{1}{2s} \exp\left(-\frac{|x-\mu|}{s}\right)$$

Args:

```
bi.dist.laplace(
loc=0.0,
scale=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *loc* (jnp.ndarray): Location parameter of the Laplace distribution.
- *sample* (jnp.ndarray): Scale parameter of the Laplace distribution. Must be positive.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.

- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Laplace distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the Laplace distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.laplace(loc=0.0, scale=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#laplace

Left Truncated

Samples from a left-truncated distribution.

A left-truncated distribution is a probability distribution obtained by restricting the support of another distribution to values greater than a specified lower bound. This is useful when dealing with data that is known to be greater than a certain value.

$$f(x) = \begin{cases} \dfrac{p(x)}{P(X > \text{low})}, & x > \text{low}, \\ 0, & \text{otherwise}, \end{cases}$$

Args:

```
bi.dist.left_truncated_distribution(
base_dist,
low=0.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

base_dist: The base distribution to truncate. Must be univariate and have real support. low: The lower truncation bound. Values less than this are excluded from the distribution.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI LeftTruncatedDistribution distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the LeftTruncatedDistribution distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#lefttruncateddistribution

---

Levy

Samples from a Levy distribution.

The probability density function is given by,

$$f(x \mid \mu, c) = \sqrt{\frac{c}{2\pi(x-\mu)^3}} \exp\left(-\frac{c}{2(x-\mu)}\right), \qquad x > \mu$$

where μ is the location parameter and c is the scale parameter.

Args:

```
bi.dist.levy(
loc,
scale,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *loc* (jnp.ndarray): Location parameter.
- *sample* (jnp.ndarray): Scale parameter.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI Levy distribution object: When `sample=False` (for model building). JAX array: When `sample=True` (for direct sampling). BI distribution object: When `create_obj=True` (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.levy(loc=0.0, scale=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#levy

Lewandowski Kurowicka Joe (LKJ)

The LKJ distribution is controlled by the concentration parameter η to make the probability of the correlation matrix M proportional to $\det(M)^{\eta-1}$. When $\eta = 1$, the distribution is uniform over correlation matrices. When $\eta > 1$, the distribution favors samples with large determinants. When $\eta < 1$, the distribution favors samples with small determinants.

$$P(M) \propto |\det(M)|^{\eta-1}$$

Args:

```
bi.dist.lkj(
dimension,
concentration=1.0,
sample_method='onion',
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

dimension (int): The dimension of the correlation matrices.

concentration (ndarray): The concentration/shape parameter of the distribution (often referred to as eta). Must be positive.

sample_method (str): Either “cvine” or “onion”. Both methods are proposed in [1] and offer the same distribution over correlation matrices. But they are different in how to generate samples. Defaults to “onion”.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI LKJ distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the LKJ distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.lkj(dimension=2, concentration=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#lkj

LKJ Cholesky

The LKJ (Leonard-Kjærgaard-Jørgensen) Cholesky distribution is a family of distributions on symmetric matrices, often used as a prior for the Cholesky decomposition of a symmetric matrix. It is particularly useful in Bayesian inference for models with covariance structure. $$

Args:

```
bi.dist.lkj_cholesky(
dimension,
concentration=1.0,
sample_method='onion',
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

dimension (int): The dimension of the correlation matrices.

concentration (float): A parameter controlling the concentration of the distribution around the identity matrix. Higher values indicate greater concentration. Must be greater than 1.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.

- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Attributes: concentration (float): The concentration parameter.

Log-Normal

The LogNormal distribution is a probability distribution defined for positive real-valued random variables, parameterized by a location parameter (loc) and a scale parameter (scale). It arises when the logarithm of a random variable is normally distributed.

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(log(x)-\mu)^2}{2\sigma^2}}$$

Args:

```
bi.dist.log_normal(
loc=0.0,
scale=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *loc* (float): Location parameter.
- *sample* (float): Scale parameter.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI LogNormal distribution object (for model building). JAX array of samples drawn from the LogNormal distribution (for direct sampling). The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.log_normal(loc=0.0, scale=1.0, sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#lognormal

---

Log-Uniform

Samples from a LogUniform distribution.

The LogUniform distribution is defined over the positive real numbers and is the result of applying an exponential transformation to a uniform distribution over the interval [low, high]. It is often used when modeling parameters that must be positive.

$$f(x) = \frac{1}{(high - low)\log(high/low)} \text{ for } low \leq x \leq high$$

Args:

```
bi.dist.log_uniform(
low,
high,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

low (jnp.ndarray): The lower bound of the uniform distribution's log-space. Must be positive.

high (jnp.ndarray): The upper bound of the uniform distribution's log-space. Must be positive.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.

- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI LogUniform distribution object (for model building) when `sample=False`.

JAX array of samples drawn from the LogUniform distribution (for direct sampling) when `sample=True`.

The raw BI distribution object (for advanced use cases) when `create_obj=True`.

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.log_uniform(low=0.1, high=10.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#loguniform

Logistic

Samples from a Logistic distribution.

The Logistic distribution is a continuous probability distribution defined by two parameters: location and scale. It is often used to model growth processes and is closely related to the normal distribution.

$$f(x) = \frac{1}{s} \exp\left(-\frac{(x-\mu)}{s}\right)$$

Args:

```
bi.dist.logistic(
loc=0.0,
scale=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *loc* (jnp.ndarray or float): The location parameter, specifying the median of the distribution. Defaults to 0.0.
- *sample* (jnp.ndarray or float): The scale parameter, which determines the spread of the distribution. Must be positive. Defaults to 1.0.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI Logistic distribution object (for model building) when `sample=False`. JAX array of samples drawn from the Logistic distribution (for direct sampling) when `sample=True`. The raw BI distribution object (for advanced use cases) when `create_obj=True`.

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.logistic(loc=0.0, scale=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#logistic

Low Rank Multivariate Normal

Represents a multivariate normal distribution with a low-rank covariance structure.

$$p(x) = \frac{1}{\sqrt{(2\pi)^K |\Sigma|}} \exp\left(-\tfrac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

where:

- x is a vector of observations.
- μ is the mean vector.
- Σ is the covariance matrix, represented in a low-rank form.

Parameters: - *loc* (jnp.ndarray): Mean vector.

cov_factor (jnp.ndarray): Matrix used to construct the covariance matrix.

cov_diag (jnp.ndarray): Diagonal elements of the covariance matrix.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Example Usage:

from BI import bi m = bi('cpu') event_size = 100 # Our distribution has 100 dimensions rank = 5
m.dist.low_rank_multivariate_normal(- *loc*=m.dist.normal(0,1, shape = (event_size,), sample=True)*2, cov_factor=m.dist.normal(0,1, shape = (event_size, rank), sample=True), cov_diag=jnp.exp(m.dist.normal(0,1, shape = (event_size,), sample=True))* 0.1, sample=True)

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#lowrankmultivariatenormal

```
bi.dist.low_rank_multivariate_normal(
loc,
cov_factor,
cov_diag,
validate_args=None,
name='x',
obs=None,
```

```
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

Lower Truncated Power Law

Lower truncated power law distribution with α index.

The probability density function (PDF) is given by:

$$f(x;\alpha,a) = (-\alpha-1)a^{-\alpha-1}x^{-\alpha}, \qquad x \geq a, \qquad \alpha < -1,$$

where a is the lower bound.

Args:

```
bi.dist.lower_truncated_power_law(
alpha,
low,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

alpha (jnp.ndarray): index of the power law distribution. Must be less than -1. low (jnp.ndarray): lower bound of the distribution. Must be greater than 0.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI LowerTruncatedPowerLaw distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the LowerTruncatedPowerLaw distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.lower_truncated_power_law(alpha=-2.0, low=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#lowertruncatedpowerlaw

---

Matrix Normal

Samples from a Matrix Normal distribution, which is a multivariate normal distribution over matrices. The distribution is characterized by a location matrix and two lower triangular matrices that define the correlation structure. The distribution is related to the multivariate normal distribution in the following way. If X $MN(loc, U, V)$ then $vec(X)$ $MVN(vec(loc), kron(V, U))$.

$$p(x) = \frac{1}{2\pi^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Args:

```
bi.dist.matrix_normal(
loc,
scale_tril_row,
scale_tril_column,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *loc* (array_like): Location of the distribution.

scale_tril_row (array_like): Lower cholesky of rows correlation matrix.

scale_tril_column (array_like): Lower cholesky of columns correlation matrix.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI MatrixNormal distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the MatrixNormal distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
import jax.numpy as jnp
m = bi('cpu')
n_rows, n_cols = 3, 4

- *loc* = jnp.zeros((n_rows, n_cols))
U_row_cov = jnp.array([[1.0, 0.5, 0.2],
```

```
[0.5, 1.0, 0.3],
[0.2, 0.3, 1.0]])
scale_tril_row = jnp.linalg.cholesky(U_row_cov)

V_col_cov = jnp.array([[2.0, -0.8, 0.1, 0.4],
[-0.8, 2.0, 0.2, -0.2],
[0.1, 0.2, 2.0, 0.0],
[0.4, -0.2, 0.0, 2.0]])

# The argument passed to the distribution is its Cholesky factor
scale_tril_column = jnp.linalg.cholesky(V_col_cov)

m.dist.matrix_normal(
oc=loc,
scale_tril_row=scale_tril_row,
scale_tril_column=scale_tril_column,
sample=True
)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#matrixnormal_lowercase

---

A marginalized finite mixture of component distributions.

This distribution represents a mixture of component distributions, where the mixing weights are determined by a Categorical distribution. The resulting distribution can be either a MixtureGeneral (when component distributions are a list) or a MixtureSameFamily (when component distributions are a single distribution).

$$p(x) = \sum_{i=1}^{K} w_i p_i(x)$$

Args:

```
bi.dist.mixture(
mixing_distribution,
component_distributions,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Mixture distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the Mixture distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from jax import random
import BI as pyro
m = pyro.distributions.Mixture(
pyro.distributions.Categorical(torch.ones(2)),
[pyro.distributions.Normal(0, 1), pyro.distributions.Normal(2, 1)]
)
samples = m.sample(sample_shape=(10,))
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#mixture

---

Mixture General

A finite mixture of component distributions from different families.

- mixing_distribution: A :class:`~numpyro.distributions.Categorical` specifying the weights for each mixture component. The size of this distribution specifies the number of components in the mixture.
- component_distributions: A list of `mixture_size` :class:`~numpyro.distributions.Distribution` objects.
- support: A :class:`~numpyro.distributions.constraints.Constraint` object specifying the support of the mixture distribution. If not provided, the support will be inferred from the component distributions.

The probability density function (PDF) of a MixtureGeneral distribution is given by:

$$p(x) = \sum_{i=1}^{K} \pi_i p_i(x)$$

where:

- K is the number of components in the mixture.
- π_i is the mixing weight for the i-th component, such that $\sum_{i=1}^{K} \pi_i = 1$.
- $p_i(x)$ is the probability density function of the i-th component distribution.

Parameters:

- **mixing_distribution**: A `Categorical` distribution representing the mixing weights.
- **component_distributions**: A list of distributions representing the components of the mixture.
- **sample (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- **seed (int, optional)**: An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- **obs (jnp.ndarray, optional)**: The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- **name (str, optional)**: The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- **When `sample=False`**: A BI MixtureGeneral distribution object (for model building).
- **When `sample=True`**: A JAX array of samples drawn from the MixtureGeneral distribution (for direct sampling).
- **When `create_obj=True`**: The raw BI distribution object (for advanced use cases).

Example Usage:

from BI import bi m = bi('cpu') m.dist.mixture_general(mixing_distribution=m.dist.categorical(probs=jnp.arr
0.7]), create_obj = True),
component_distributions=[m.dist.normal(loc=0.0, scale=1.0, create_obj=True),m.dist.normal
(loc=0.0, scale=1.0, create_obj=True)], sample = True)

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#mixturegeneral

```
bi.dist.mixture_general(
mixing_distribution,
component_distributions,
support=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

Finite mixture of component distributions from the same family.

This mixture only supports a mixture of component distributions that are all of the same family. The different components are specified along the last batch dimension of the input `component_distribution`. If you need a mixture of distributions from different families, use the more general implementation in :class:`~numpyro.distributions.MixtureGeneral`.

$$p(x) = \sum_{k=1}^{K} w_k p_k(x)$$

where:

- K is the number of mixture components.
- w_k is the mixing weight for component k.
- $p_k(x)$ is the probability density function (PDF) of the k-th component distribution.

Args:

```
bi.dist.mixture_same_family(
mixing_distribution,
component_distribution,
validate_args=None,
name='x',
```

```
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- **Distribution Args**:
- **mixing_distribution**: A `Categorical` distribution representing the mixing weights.
- **component_distributions**: A list of distributions representing the components of the mixture.
- **Sampling / Modeling Args**:
- `shape` (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- `event` (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- `mask` (jnp.ndarray, bool): Optional boolean array to mask observations.
- `create_obj` (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- **sample (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- **seed (int, optional)**: An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- **obs (jnp.ndarray, optional)**: The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- **name (str, optional)**: The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- **When `sample=False`**: A BI MixtureSameFamily distribution object (for model building).
- **When `sample=True`**: A JAX array of samples drawn from the MixtureSameFamily distribution (for direct sampling).
- **When `create_obj=True`**: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.mixture_same_family(
mixing_distribution=m.dist.categorical(probs=jnp.array([0.3, 0.7]), create_obj = True),
component_distribution=m.dist.normal(loc=0.0, scale=1.0, shape = (2,), create_obj=True),
sample = True
)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#mixture-same-family

---

Multinomial

Samples from a Multinomial distribution, which models the probability of different outcomes in a sequence of independent trials, each with a fixed number of trials and a fixed set of possible outcomes. It generalizes the binomial distribution to multiple categories.

$$P(X = x) = \frac{n!}{x_1!x_2!\cdots x_k!}p_1^{x_1}p_2^{x_2}\cdots p_k^{x_k}$$

Args:

```
bi.dist.multinomial(
total_count=1,
probs=None,
logits=None,
total_count_max=None,
validate_args=None,
name='x',
```

```
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

total_count (int or jnp.ndarray): The number of trials.

- *probs* (jnp.ndarray, optional): Event probabilities. Must sum to 1.

logits (jnp.ndarray, optional): Event log probabilities.

total_count_max (int, optional): An optional integer providing an upper bound on `total_count`. This is used for performance optimization with `lax.scan` when `total_count` is a dynamic JAX tracer, helping to avoid recompilation.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool, optional): Optional boolean array to mask observations.
- *create_obj* (bool, optional): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Multinomial distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the Multinomial distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.multinomial(total_count=10, probs=jnp.array([0.2, 0.3, 0.5]), sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#multinomial

---

Multinomial Logits

Samples from a MultinomialLogits distribution.

This distribution represents the probability of observing a specific outcome from a multinomial experiment, given the logits for each outcome. The logits are the natural logarithm of the odds of each outcome.

$$P(k|\ ) = \frac{n!}{k!(n-k)!} \prod_{i=1}^{k} \pi_i$$

Args:

```
bi.dist.multinomial_logits(
logits,
total_count=1,
total_count_max=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
```

```
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

logits (jnp.ndarray): Logits for each outcome. Must be at least one-dimensional.

total_count (jnp.ndarray): The total number of trials.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI MultinomialLogits distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the MultinomialLogits distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.multinomial_logits(logits=jnp.array([1.0, 0.5], dtype=jnp.float32), total_count=jnp.ar
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#multinomiallogits

---

Multinomial Probs

Samples from a Multinomial distribution.

The Multinomial distribution models the number of times each of several discrete outcomes occurs in a fixed number of trials. Each trial independently results in one of several outcomes, and each outcome has a probability of occurring.

$$P(X = x) = \frac{n!}{x_1!x_2!\cdots x_k!}p_1^{x_1}p_2^{x_2}\cdots p_k^{x_k}$$

where:

- n is the total number of trials.
- x is a vector of counts for each outcome.
- p is a vector of probabilities for each outcome. $$

Args:

```
bi.dist.multinomial_probs(
probs,
total_count=1,
total_count_max=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
```

```
shape=(),
event=0,
create_obj=False,
)
```

- *probs* (jnp.ndarray): Vector of probabilities for each outcome. Must sum to 1. total_count (jnp.ndarray): The number of trials.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI MultinomialProbs distribution object (for model building). JAX array of samples drawn from the MultinomialProbs distribution (for direct sampling). The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.multinomial_probs(probs=jnp.array([0.2, 0.3, 0.5]), total_count=10, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#multinomialprobs

---

Multivariate Normal

The Multivariate Normal distribution, also known as the Gaussian distribution in multiple dimensions, is a probability distribution that arises frequently in statistics and machine learning. It is defined by its mean vector and covariance matrix, which describe the central tendency and spread of the distribution, respectively.

$$p(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

where: - x is a n-dimensional vector of random variables. - μ is the mean vector. - Σ is the covariance matrix.

Args:

```
bi.dist.multivariate_normal(
loc=0.0,
covariance_matrix=None,
precision_matrix=None,
scale_tril=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *loc* (tuple): The mean vector of the distribution.

covariance_matrix (jnp.ndarray, optional): The covariance matrix of the distribution. Must be positive definite.

precision_matrix (jnp.ndarray, optional): The precision matrix (inverse of the covariance matrix) of the distribution. Must be positive definite.

scale_tril (jnp.ndarray, optional): The lower triangular Cholesky decomposition of the covariance matrix.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI MultivariateNormal distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the MultivariateNormal distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.multivariate_normal(
- *loc*=jnp.array([1.0, 0.0, -2.0]),
covariance_matrix=jnp.array([[ 2.0,  0.7, -0.3],
[ 0.7,  1.0,  0.5],
[-0.3,  0.5,  1.5]]),
sample=True
)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#multivariate-normal

---

Multivariate Student's t

The Multivariate Student's t distribution is a generalization of the Student's t distribution to multiple dimensions. It is a heavy-tailed distribution that is often used to model data that is not normally distributed.

$$p(x) = \frac{1}{B(df/2, n/2)} \frac{\Gamma(df/2 + n/2)}{\Gamma(df/2)} \left(1 + \frac{(x-\mu)^T \Sigma^{-1} (x-\mu)}{df}\right)^{-(df+n)/2}$$

Args:

```
bi.dist.multivariate_student_t(
df,
loc=0.0,
scale_tril=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
```

```
create_obj=False,
)
```

- *df* (jnp.ndarray): Degrees of freedom, must be positive.
- *loc* (jnp.ndarray): Location vector, representing the mean of the distribution. scale_tril (jnp.ndarray): Lower triangular matrix defining the scale.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI MultivariateStudentT distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the MultivariateStudentT distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
import jax.numpy as jnp
m = bi('cpu')
m.dist.multivariate_student_t(
df = 2,
- *loc*=jnp.array([1.0, 0.0, -2.0]),
scale_tril=jnp.linalg.cholesky(
jnp.array([[ 2.0,  0.7, -0.3],
[ 0.7,  1.0,  0.5],
[-0.3,  0.5,  1.5]])),
sample=True
)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#multivariatestudentt

---

Negative Binomial

The NegativeBinomial distribution models the number of failures before the first success in a sequence of independent Bernoulli trials. It is characterized by two parameters: 'total_count' (r) and 'probs' or 'logits' (p).

$$P(k) = \binom{k+r-1}{r-1} p^r (1-p)^k$$

Args:

```
bi.dist.negative_binomial(
total_count,
probs=None,
logits=None,
validate_args=None,
name='x',
obs=None,
mask=None,
```

```
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

total_count (jnp.ndarray): The total number of events.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI NegativeBinomial distribution object (for model building). JAX array of samples drawn from the NegativeBinomial distribution (for direct sampling). The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.negative_binomial(total_count=5.0,probs = jnp.array([0.2, 0.3, 0.5]), sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#negativebinomial

---

Negative Binomial Logits

Samples from a Negative Binomial Logits distribution.

The Negative Binomial Logits distribution is a generalization of the Negative Binomial distribution where the parameter 'r' (number of successes) is expressed as a function of a logit parameter. This allows for more flexible modeling of count data.

$$P(k) = \frac{e^{-n \cdot \text{softplus}(x)} \cdot \text{softplus}(-x)^k}{k!}$$

Args:

```
bi.dist.negative_binomial_logits(
total_count,
logits,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

total_count (jnp.ndarray): The parameter controlling the shape of the distribution. Represents the total number of trials.

logits (jnp.ndarray): The log-odds parameter. Related to the probability of success.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

Negative Binomial Logits: A BI Negative Binomial Logits distribution object (for model building).

jnp.ndarray: A JAX array of samples drawn from the Negative Binomial Logits distribution (for direct sampling).

Negative Binomial Logits: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.negative_binomial_logits(total_count=5.0, logits=0.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#Negative Binomial Logits

---

Negative Binomial with probabilities.

The Negative Binomial distribution models the number of failures before the first success in a sequence of independent Bernoulli trials. It is characterized by two parameters: 'concentration' (r) and 'rate' (p). In this implementation, the 'concentration' parameter is derived from 'total_count' and the 'rate' parameter is derived from 'probs'.

$$P(k) = \binom{k+r-1}{r-1} p^r (1-p)^k$$

Args:

```
bi.dist.negative_binomial_probs(
total_count,
probs,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

total_count (jnp.ndarray): A numeric vector, matrix, or array representing the parameter.

- *probs* (jnp.ndarray): A numeric vector representing event probabilities. Must sum to 1.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.

- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI NegativeBinomialProbs distribution object (for model building). JAX array of samples drawn from the NegativeBinomialProbs distribution (for direct sampling). The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.negative_binomial_probs(total_count=10.0, probs = jnp.array([0.2, 0.3, 0.5]), sample=
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#negativebinomialprobs

---

Normal

Samples from a Normal (Gaussian) distribution.

The Normal distribution is characterized by its mean (loc) and standard deviation (scale). It's a continuous probability distribution that arises frequently in statistics and probability theory.

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Args:

```
bi.dist.normal(
loc=0.0,
scale=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *loc* (jnp.ndarray): The mean of the distribution.
- *sample* (jnp.ndarray): The standard deviation of the distribution.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Normal distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the Normal distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.normal(loc=0.0, scale=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#normal

Ordered Logistic

A categorical distribution with ordered outcomes. This distribution represents the probability of an event falling into one of several ordered categories, based on a predictor variable and a set of cutpoints. The probability of an event falling into a particular category is determined by the number of categories above it.

$$P(Y = k) = \begin{cases} 1 & \text{if } k = 0 \\ \frac{1}{k} & \text{if } k > 0 \end{cases}$$

Args:

```
bi.dist.ordered_logistic(
predictor,
cutpoints,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- predictor (jnp.ndarray): Prediction in real domain; typically this is output of a linear model.
- cutpoints (jnp.ndarray): Positions in real domain to separate categories.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI OrderedLogistic distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the OrderedLogistic distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.ordered_logistic(predictor=jnp.array([0.2, 0.5, 0.8]), cutpoints=jnp.array([-1.0, 0.0
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#orderedlogistic

---

Pareto

Samples from a Pareto distribution.

The Pareto distribution is a power-law probability distribution that is often used to model income, wealth, and the size of cities. It is defined by two parameters: alpha (shape) and scale.

$$f(x) = \frac{\alpha \cdot \text{scale}^{\alpha}}{x^{\alpha+1}} \text{ for } x \geq \text{scale}$$

Args:

```
bi.dist.pareto(
scale,
alpha,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *sample* (jnp.ndarray or float): Scale parameter of the Pareto distribution. Must be positive.

alpha (jnp.ndarray or float): Shape parameter of the Pareto distribution. Must be positive.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.

- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Pareto distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the Pareto distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.pareto(scale=2.0, alpha=3.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#pareto

Poisson

Creates a Poisson distribution, a discrete probability distribution that models the number of events occurring in a fixed interval of time or space if these events occur with a known average rate and independently of the time since the last event.

$$\text{rate}^k \frac{e^{-\text{rate}}}{k!}$$

Args:

```
bi.dist.poisson(
rate,
is_sparse=False,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *rate* (jnp.ndarray): The rate parameter, representing the average number of events. is_sparse (bool, optional): Indicates whether the `rate` parameter is sparse. If `True`, a specialized sparse sampling implementation is used, which can be more efficient for models with many zero-rate components (e.g., zero-inflated models). Defaults to `False`.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Poisson distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the Poisson distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.poisson(rate=2.0, sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#poisson

---

Projected Normal

This distribution over directional data is qualitatively similar to the von Mises and von Mises-Fisher distributions, but permits tractable variational inference via reparametrized gradients.

$$p(x) = \frac{1}{Z} \exp\left(-\frac{1}{2\sigma^2}||x-\mu||^2\right)$$

Args:

```
bi.dist.projected_normal(
concentration,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

concentration (jnp.ndarray): The concentration parameter, representing the direction towards which the samples are concentrated. Must be a JAX array with at least one dimension.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI ProjectedNormal distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the ProjectedNormal distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.projected_normal(concentration=jnp.array([1.0, 3.0, 2.0]), sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#projectednormal

---

Relaxed Bernoulli

The Relaxed Bernoulli distribution is a continuous relaxation of the discrete Bernoulli distribution. It's useful for variational inference and other applications where a differentiable approximation of the Bernoulli is needed. The probability density function (PDF) is defined as:

$$p(x) = \frac{1}{2}\left(1 + \tanh\left(\frac{x - \beta \log(\frac{p}{1-p})}{1}\right)\right)$$

Args:

```
bi.dist.relaxed_bernoulli(
temperature,
probs=None,
logits=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

temperature (float): The temperature parameter.

- *probs* (jnp.ndarray, optional): The probability of success. Must be in the interval `[0, 1]`. Only one of `probs` or `logits` can be specified.

logits (jnp.ndarray, optional): The log-odds of success. Only one of `probs` or `logits` can be specified.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI RelaxedBernoulli distribution object (for model building) when `sample=False`. A JAX array of samples drawn from the RelaxedBernoulli distribution (for direct sampling) when `sample=True`. The raw BI distribution object (for advanced use cases) when `create_obj=True`.

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.relaxed_bernoulli(temperature=1.0, probs = jnp.array([0.2, 0.3, 0.5]), sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#relaxedbernoulli

---

Relaxed Bernoulli Logits

Represents a relaxed version of the Bernoulli distribution, parameterized by logits and a temperature. The temperature parameter controls the sharpness of the distribution. The distribution is defined by transforming the output of a Logistic distribution through a sigmoid function.

$$P(x) = \sigma\left(\frac{x}{\text{temperature}}\right)$$

Args:

```
bi.dist.relaxed_bernoulli_logits(
temperature,
logits,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

temperature (jnp.ndarray): The temperature parameter, must be positive. logits (jnp.ndarray): The logits parameter.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

RelaxedBernoulliLogits: A BI RelaxedBernoulliLogits distribution object (for model building). jnp.ndarray: A JAX array of samples drawn from the RelaxedBernoulliLogits distribution (for direct sampling). RelaxedBernoulliLogits: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.relaxed_bernoulli_logits(temperature=1.0, logits=0.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#relaxed-bernoulli-logits

---

Right Truncated

Samples from a right-truncated distribution.

This distribution truncates the base distribution at a specified high value. Values greater than `high` are discarded, effectively creating a distribution that is only supported up to that point. This is useful for modeling data where observations are only possible within a certain range.

The probability density function (PDF) of the truncated distribution is:

$$f_{\text{trunc}}(x) = \frac{f_{\text{base}}(x)}{F_{\text{base}}(\text{high})} \quad \text{for } x \leq \text{high}$$

where $f_{\text{base}}(x)$ is the PDF of the base distribution and $F_{\text{base}}(\text{high})$ is the cumulative distribution function (CDF) of the base distribution evaluated at `high`.

where $f(x)$ is the probability density function (PDF) of the base distribution and $P(X \leq high)$ is the cumulative distribution function (CDF) of the base distribution evaluated at `high`. $$

Args:

```
bi.dist.right_truncated_distribution(
base_dist,
high=0.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

base_dist: The base distribution to truncate. Must be a univariate distribution with real support.

high (float, jnp.ndarray, optional): The upper truncation point. The support of the new distribution is $(-\infty, \text{high}]$. Defaults to 0.0.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI RightTruncatedDistribution distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the RightTruncatedDistribution distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.right_truncated_distribution(base_dist = m.dist.normal(0,1, create_obj = True), high=
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#righttruncateddistribution

---

Sine Bivariate Von Mises

A unimodal distribution for two dependent angles on the 2-torus ($S^1 \otimes S^1$), which is useful for modeling coupled angles like torsion angles in peptide chains. [1]

The probability density function is given by:

$$C^{-1} \exp(\kappa_1 \cos(x_1 - \mu_1) + \kappa_2 \cos(x_2 - \mu_2) + \rho \sin(x_1 - \mu_1) \sin(x_2 - \mu_2))$$

where the normalization constant C is:

$$C = (2\pi)^2 \sum_{i=0}^{\infty} \binom{2i}{i} \left(\frac{\rho^2}{4\kappa_1\kappa_2} \right)^i I_i(\kappa_1) I_i(\kappa_2)$$

Here, $I_i(\cdot)$ is the modified Bessel function of the first kind, μ's are the locations, κ's are the concentrations, and ρ represents the correlation between the angles x_1 and x_2.

Args:

```
bi.dist.sine_bivariate_vonmises(
phi_loc,
psi_loc,
phi_concentration,
psi_concentration,
correlation=None,
weighted_correlation=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *phi_loc* (jnp.ndarray): The location parameter for the first angle (phi).
- *psi_loc* (jnp.ndarray): The location parameter for the second angle (psi).
- *phi_concentration* (jnp.ndarray): The concentration parameter for the first angle (phi). Must be positive.

- *psi_concentration* (jnp.ndarray): The concentration parameter for the second angle (psi). Must be positive.
- *correlation* (jnp.ndarray, optional): The correlation parameter between the two angles. One of `correlation` or `weighted_correlation` must be specified.
- *weighted_correlation* (jnp.ndarray, optional): An alternative correlation parameter. One of `correlation` or `weighted_correlation` must be specified.
- *validate_args* (bool, optional): Whether to enable validation of distribution parameters. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *mask* (jnp.ndarray, bool, optional): Optional boolean array to mask observations. If provided, events with a `True` mask will be conditioned on `obs`, while the remaining events will be treated as latent variables. Defaults to `None`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *shape* (tuple, optional): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int, optional): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *create_obj* (bool, optional): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`. Defaults to `False`.

Returns:

BI.primitives.Messenger: A BI sample site object when used in a model context (`sample=False`). jnp.ndarray: A JAX array of samples drawn from the SineBivariateVonMises distribution (for direct sampling, `sample=True`). numpyro.distributions.Distribution: The raw BI distribution object (if `create_obj=True`).

Example Usage:

```
from BI import bi
import jax.numpy as jnp
m = bi('cpu')

# Direct sampling
samples = m.dist.sine_bivariate_vonmises(
phi_loc=0.0,
psi_loc=jnp.pi,
phi_concentration=1.0,
psi_concentration=1.0,
correlation=0.5,
sample=True,
- *shape*=(10,)
)

# Usage within a model
def my_model():
angles = m.dist.sine_bivariate_vonmises(
phi_loc=0.0,
psi_loc=0.0,
phi_concentration=2.0,
psi_concentration=2.0,
weighted_correlation=0.9,
name='angles'
)
# ... rest of the model
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#sinebivariatevonmises

---

Sine-skewing

Sine-skewing [1] is a procedure for producing a distribution that breaks pointwise symmetry on a torus distribution. The new distribution is called the Sine Skewed X distribution, where X is the name of the (symmetric) base distribution. Torus distributions are distributions with support on products of circles (i.e., $\otimes S^1$ where $S^1 = [-pi, pi)$). So, a 0-torus is a point, the 1-torus is a circle, and the 2-torus is commonly associated with the donut shape.

.. note: This distribution is available in BI: https://num.pyro.ai/en/stable/distributions.html#sineskewed

Parameters:

- **base_dist:** Base density on a d-dimensional torus. Supported base distributions include: 1D :class:`~numpyro.distributions.VonMises`, :class:`~numnumpyro.distributions.SineBivariateVonM 1D :class:`~numpyro.distributions.ProjectedNormal`, and :class:`~numpyro.distributions.Uniform` (-pi, pi).
- **skewness:** Skewness of the distribution.
- **sample (bool, optional):** A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- **seed (int, optional):** An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- **obs (jnp.ndarray, optional):** The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- **name (str, optional):** The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

PDF:

The probability density function (PDF) of the Sine Skewed X distribution is not explicitly defined here, but it is derived from the base distribution and the skewness parameter.

Example Usage:

from num.pyro import distributions as dist import num.pyro as pyro import num.numpy as np

m = pyro.distributions.Normal(loc=0.0, scale=1.0) skewness = np.array([0.5, 0.5]) sine_skewed = dist.SineSkewed(base_dist=m, skewness=skewness) samples = sine_skewed.sample((1000,))

```
bi.dist.sine_skewed(
base_dist: numpyro.distributions.distribution.Distribution,
skewness,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

SoftLaplace

Samples from a SoftLaplace distribution.

This distribution is a smooth approximation of a Laplace distribution, characterized by its log-convex density. It offers Laplace-like tails while being infinitely differentiable, making it suitable for HMC and Laplace approximation.

$$f(x) = \log\left(\frac{2}{\pi}\right) - \log(\text{scale}) - \log\left(e^{\frac{x-\text{loc}}{\text{scale}}} + e^{-\frac{x-\text{loc}}{\text{scale}}}\right)$$

Args:

```
bi.dist.soft_laplace(
loc,
scale,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
```

```
create_obj=False,
)
```

- *loc*: Location parameter. scale: Scale parameter. $$

Args:

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI SoftLaplace distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the SoftLaplace distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.soft_laplace(loc=0.0, scale=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#softlaplace

Student's t

The Student's t-distribution is a probability distribution that arises in hypothesis testing involving the mean of a normally distributed population when the population standard deviation is unknown. It is similar to the normal distribution, but has heavier tails, making it more robust to outliers.

$$f(x) = \frac{1}{\Gamma(\nu/2)\sqrt{\nu\pi}} \left(1 + \frac{x^2}{\nu}\right)^{-(\nu+1)/2}$$

Args:

```
bi.dist.student_t(
df,
loc=0.0,
scale=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

df (jnp.ndarray): Degrees of freedom, must be positive. - *loc* (jnp.ndarray): Location parameter, defaults to 0.0. - *sample* (jnp.ndarray): Scale parameter, defaults to 1.0.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI StudentT distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the StudentT distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.student_t(df = 2, loc=0.0, scale=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#studentt

Truncated Cauchy

The Cauchy distribution, also known as the Lorentz distribution, is a continuous probability distribution that appears frequently in various areas of mathematics and physics. It is characterized by its heavy tails, which extend to infinity. The truncated version limits the support of the Cauchy distribution to a specified interval.

$$f(x) = \frac{1}{\pi \cdot c \cdot (1 + ((x - b)/c)^2)} \text{ for } a < x < b$$

Args:

```
bi.dist.truncated_cauchy(
loc=0.0,
scale=1.0,
low=None,
high=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *loc* (float): Location parameter of the Cauchy distribution.
- *sample* (float): Scale parameter of the Cauchy distribution.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.

- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI TruncatedCauchy distribution object (for model building) when `sample=False`.

JAX array of samples drawn from the TruncatedCauchy distribution (for direct sampling) when `sample=True`.

The raw BI distribution object (for advanced use cases) when `create_obj=True`.

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.truncated_cauchy(loc=0.0, scale=1.0, sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#truncatedcauchy

---

Truncated

Samples from a Truncated Distribution.

This distribution represents a base distribution truncated between specified lower and upper bounds. The truncation modifies the probability density function (PDF) of the base distribution, effectively removing observations outside the defined interval.

$$p(x) = \frac{p(x)}{P(\text{lower} \leq x \leq \text{upper})}$$

Args:

```
bi.dist.truncated_distribution(
base_dist,
low=None,
high=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

base_dist: The base distribution to be truncated. This should be a univariate distribution. Currently, only the following distributions are supported: Cauchy, Laplace, Logistic, Normal, and StudentT.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI TruncatedDistribution distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the TruncatedDistribution distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.truncated_distribution(base_dist = m.dist.normal(0,1, create_obj = True), high=1, low
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#truncateddistribution

---

Truncated Normal

The Truncated Normal distribution is a normal distribution truncated to a specified interval. It is defined by its location (`loc`), scale (`scale`), lower bound (`low`), and upper bound (`high`).

$$f(x) = \frac{p(x)}{\alpha}, \quad x \in [\text{low}, \text{high}]$$

where

$$p(x) = \frac{1}{\text{scale}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\text{loc}}{\text{scale}}\right)^2\right),$$

and

$$\alpha = \int_{\text{low}}^{\text{high}} p(x)\, dx.$$

Args:

```
bi.dist.truncated_normal(
loc=0.0,
scale=1.0,
low=None,
high=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *loc* (float): The location parameter of the normal distribution.
- *sample* (float): The scale parameter of the normal distribution.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI TruncatedNormal distribution object (for model building). JAX array of samples drawn from the TruncatedNormal distribution (for direct sampling). The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.truncated_normal(loc=0.0, scale=1.0, sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#truncatednormal_lowercase

---

Truncated PolyaGamma

Samples from a Truncated PolyaGamma distribution.

This distribution is a truncated version of the PolyaGamma distribution, defined over the interval [0, truncation_point]. It is often used in Bayesian non-parametric models.

$$p(x) = \frac{1}{Z} \exp\left(\sum_{n=0}^{N}\left(\log(2n+1) - 1.5\log(x) - \frac{(2n+1)^2}{4x}\right)\right)$$

Args:

```
bi.dist.truncated_polya_gamma(
batch_shape=(),
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

batch_shape (tuple): The shape of the batch dimension.

- *event* (int): The number of batch dimensions to reinterpret as event dimensions.
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Truncated PolyaGamma distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the Truncated PolyaGamma distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.truncated_polya_gamma(batch_shape=(), sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#truncatedpolygammadistribution

---

Two Sided Truncated

This distribution truncates a base distribution between two specified lower and upper bounds.

$$f(x) = \begin{cases} \dfrac{p(x)}{P(\text{low} \leq X \leq \text{high})}, & \text{if low} \leq x \leq \text{high}, \\ 0, & \text{otherwise.} \end{cases}$$

where $p(x)$ is the probability density function of the base distribution.

Args:

```
bi.dist.two_sided_truncated_distribution(
base_dist,
low=0.0,
high=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

base_dist: The base distribution to truncate.

low: The lower bound for truncation.

high: The upper bound for truncation.

- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI TwoSidedTruncatedDistribution distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the TwoSidedTruncatedDistribution distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#twosidedtruncateddistribution

---

Uniform

Samples from a Uniform distribution, which is a continuous probability distribution where all values within a given interval are equally likely.

$$f(x) = \frac{1}{b-a}, \text{ for } a \leq x \leq b$$

Args:

```
bi.dist.uniform(
low=0.0,
high=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

low (jnp.ndarray): The lower bound of the uniform interval.

high (jnp.ndarray): The upper bound of the uniform interval.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI Uniform distribution object (for model building) when `sample=False`.

JAX array of samples drawn from the Uniform distribution (for direct sampling) when `sample=True`.

The raw BI distribution object (for advanced use cases) when `create_obj=True`.

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.uniform(low=0.0, high=1.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#uniform

Unit

The Unit distribution is a trivial, non-normalized distribution representing the unit type. It has a single value with no data, effectively a placeholder often used in probabilistic programming for situations where no actual data is involved.

$$p(x) = 1$$

Args:

```
bi.dist.unit(
log_factor,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
```

```
create_obj=False,
)
```

log_factor (jnp.ndarray): Log factor for the unit distribution. This parameter determines the - *shape* and batch size of the distribution.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI Unit distribution object: When `sample=False` (for model building). jnp.ndarray: A JAX array of samples drawn from the Unit distribution (for direct sampling). BI Unit distribution object: When `create_obj=True` (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.unit(log_factor=jnp.ones(5), sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#unit

Weibull

Samples from a Weibull distribution.

The Weibull distribution is a versatile distribution often used to model failure rates in engineering and reliability studies. It is characterized by its shape and scale parameters.

$$f(x) = \frac{\beta}{\alpha}\left(\frac{x}{\alpha}\right)^{\beta-1} e^{-\left(\frac{x}{\alpha}\right)^{\beta}} \text{ for } x \geq 0$$

where α is the scale parameter and β is the shape parameter.

Args:

```
bi.dist.weibull(
scale,
concentration,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *sample* (jnp.ndarray): The scale parameter of the Weibull distribution. Must be positive. concentration (jnp.ndarray): The shape parameter of the Weibull distribution. Must be positive.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.

- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI Weibull distribution object (for model building) when `sample=False`. JAX array of samples drawn from the Weibull distribution (for direct sampling) when `sample=True`. The raw BI distribution object (for advanced use cases) when `create_obj=True`.

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.weibull(scale=1.0, concentration=2.0, sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#weibull

Wishart

The Wishart distribution is a multivariate distribution used to model positive definite matrices, often representing covariance matrices. It's commonly used in Bayesian statistics and machine learning, particularly in models involving covariance estimation.

$$p(X) = \frac{1}{W^{p/2}\Gamma_p(concentration/2)} |X|^{-concentration/2} \exp\left(-\frac{1}{2}\text{tr}(X^{-1}X)\right)$$

Args:

```
bi.dist.wishart(
concentration,
scale_matrix=None,
rate_matrix=None,
scale_tril=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

concentration (jnp.ndarray): Positive concentration parameter analogous to the concentration of a :class:`Gamma` distribution. The concentration must be larger than the dimensionality of the scale matrix.

scale_matrix (jnp.ndarray, optional): Scale matrix analogous to the inverse rate of a :class:`Gamma` distribution.

rate_matrix (jnp.ndarray, optional): Rate matrix anaologous to the rate of a :class:`Gamma` distribution.

scale_tril (jnp.ndarray, optional): Cholesky decomposition of the :code:`scale_matrix`.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.

- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool, optional): Optional boolean array to mask observations.
- *create_obj* (bool, optional): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI Wishart distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the Wishart distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.wishart(concentration=5.0, scale_matrix=jnp.eye(2), sample=True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#wishart

Wishart Cholesky

The Wishart distribution is a multivariate distribution used as a prior distribution for covariance matrices. This implementation represents the distribution in terms of its Cholesky decomposition.

.. rubric:: Probability Density Function

The probability density function (PDF) is given by:

PDF = (1 / ((2 * pi)^(k * (k - 1) / 2) * Gamma(k/2)) *(concentration^(k/2)* det(scale_matrix))^(-1/2) *exp(-1/2* trace(rate_matrix @ scale_matrix)))

where:

- k is the dimensionality of the covariance matrix.
- concentration is a positive concentration parameter.
- scale_matrix is the scale matrix.
- rate_matrix is the rate matrix.
- Gamma is the gamma function.

.. rubric:: Parameters

- concentration: (Tensor) Positive concentration parameter analogous to the concentration of a :class:`Gamma` distribution. The concentration must be larger than the dimensionality of the scale matrix.
- scale_matrix: (Tensor, optional) Scale matrix analogous to the inverse rate of a :class:`Gamma` distribution. If not provided, `rate_matrix` or `scale_tril` must be.
- rate_matrix: (Tensor, optional) Rate matrix anaologous to the rate of a :class:`Gamma` distribution. If not provided, `scale_matrix` or `scale_tril` must be.
- scale_tril: (Tensor, optional) Cholesky decomposition of the :code:`scale_matrix`. If not provided, `scale_matrix` or `rate_matrix` must be.
- sample (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- seed (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- obs (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.

- name (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

```
bi.dist.wishart_cholesky(
concentration,
scale_matrix=None,
rate_matrix=None,
scale_tril=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

Generic Zero Inflated

A Zero-Inflated distribution combines a base distribution with a Bernoulli distribution to model data with an excess of zero values. It assumes that each observation is either drawn from the base distribution or is a zero with probability determined by the Bernoulli distribution (the "gate"). This is useful for modeling data where zeros are more frequent than expected under a single distribution, often due to a different underlying process.

$$P(x) = \pi \cdot I(x = 0) + (1 - \pi) \cdot P_{base}(x)$$

where: - $P_{base}(x)$ is the probability density function (PDF) or probability mass function (PMF) of the base distribution. - π is the probability of generating a zero, governed by the Bernoulli gate. - $I(x = 0)$ is an indicator function that equals 1 if x=0 and 0 otherwise.

Args:

```
bi.dist.zero_inflated_distribution(
base_dist,
gate=None,
gate_logits=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

base_dist (Distribution): The base distribution to be zero-inflated (e.g., Poisson, NegativeBinomial).

gate (jnp.ndarray, optional): Probability of extra zeros (between 0 and 1).

gate_logits (jnp.ndarray, optional): Log-odds of extra zeros.

validate_args (bool, optional): Whether to validate parameter values. Defaults to None.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.

- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI ZeroInflatedDistribution distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the ZeroInflatedDistribution distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.zero_inflated_distribution(base_dist=m.dist.poisson(rate=5, create_obj = True), gate =
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#zeroinflateddistribution

---

Zero-Inflated Negative Binomial

This distribution combines a Negative Binomial distribution with a binary gate variable. Observations are either drawn from the Negative Binomial distribution with probability (1 - gate) or are treated as zero with probability 'gate'. This models data with excess zeros compared to what a standard Negative Binomial distribution would predict.

$$P(X = x) = (1 - gate) \cdot \frac{\Gamma(x + \alpha)}{\Gamma(x + \alpha + \beta)\Gamma(\alpha)} \left(\frac{\beta}{\alpha + \beta}\right)^x + gate \cdot \delta_{x,0}$$

Args:

```
bi.dist.zero_inflated_negative_binomial2(
mean,
concentration,
gate=None,
gate_logits=None,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

mean (jnp.ndarray or float): The mean of the Negative Binomial 2 distribution. concentration (jnp.ndarray or float): The concentration parameter of the Negative Binomial 2 distribution.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI ZeroInflatedNegativeBinomial2 distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the ZeroInflatedNegativeBinomial2 distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.zero_inflated_negative_binomial2(mean=2.0, concentration=3.0, gate = 0.3, sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#zeroinflatednegativebinomial2

A Zero Inflated Poisson

This distribution combines two Poisson processes: one with a rate parameter and another that generates only zeros. The probability of observing a zero is determined by the 'gate' parameter, while the probability of observing a non-zero value is governed by the 'rate' parameter of the underlying Poisson distribution.

$$P(X = k) = (1 - gate) * \frac{e^{-rate} rate^k}{k!} + gate$$

Args:

```
bi.dist.zero_inflated_poisson(
gate,
rate=1.0,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
```

```
shape=(),
event=0,
create_obj=False,
)
```

rate (jnp.ndarray): The rate parameter of the underlying Poisson distribution.

- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.
- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

BI ZeroInflatedPoisson distribution object (when `sample=False`). JAX array of samples drawn from the ZeroInflatedPoisson distribution (when `sample=True`). The raw BI distribution object (when `create_obj=True`).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.zero_inflated_poisson(gate = 0.3, rate=2.0, sample=True)
```

Wrapper of: https://num.pyro.ai/en/stable/distributions.html#zeroinflatedpoisson

---

Zero Sum Normal

Samples from a ZeroSumNormal distribution, which is a Normal distribution where one or more axes are constrained to sum to zero.

$$ZSN(\sigma) = N(0, \sigma^2(I - \tfrac{1}{n}J))\text{where} \;\; J_{ij} = 1 \;\; \text{and} n = \text{number of zero-sum axes}$$

Args:

```
bi.dist.zero_sum_normal(
scale,
event_shape,
validate_args=None,
name='x',
obs=None,
mask=None,
sample=False,
seed=0,
shape=(),
event=0,
create_obj=False,
)
```

- *sample* (array_like): Standard deviation of the underlying normal distribution before the zerosum constraint is enforced.
- *shape* (tuple): A multi-purpose argument for shaping. When `sample=False` (model building), this is used with `.expand(shape)` to set the distribution's batch shape. When `sample=True` (direct sampling), this is used as `sample_shape` to draw a raw JAX array of the given shape.

- *event* (int): The number of batch dimensions to reinterpret as event dimensions (used in model building).
- *mask* (jnp.ndarray, bool): Optional boolean array to mask observations.
- *create_obj* (bool): If True, returns the raw BI distribution object instead of creating a sample site. This is essential for building complex distributions like `MixtureSameFamily`.
- *sample* (bool, optional): A control-flow argument. If `True`, the function will directly sample a raw JAX array from the distribution, bypassing the BI model context. If `False`, it will create a `BI.sample` site within a model. Defaults to `False`.
- *seed* (int, optional): An integer used to generate a JAX PRNGKey for reproducible sampling when `sample=True`. [7] This argument has no effect when `sample=False`, as randomness is handled by BI's inference engine. Defaults to 0.
- *obs* (jnp.ndarray, optional): The observed value for this random variable. If provided, the sample site is conditioned on this value, and the function returns the observed value. If `None`, the site is treated as a latent (unobserved) random variable. Defaults to `None`.
- *name* (str, optional): The name of the sample site in a BI model. This is used to uniquely identify the random variable. Defaults to 'x'.

Returns:

- When `sample=False`: A BI ZeroSumNormal distribution object (for model building).
- When `sample=True`: A JAX array of samples drawn from the ZeroSumNormal distribution (for direct sampling).
- When `create_obj=True`: The raw BI distribution object (for advanced use cases).

Example Usage:

```
from BI import bi
m = bi('cpu')
m.dist.zero_sum_normal(scale=1.0, event_shape = (2,), sample = True)
```

Wrapper of:

https://num.pyro.ai/en/stable/distributions.html#zerosumnormal

---