multivarious.rvs for random variables

Distribution	$f_X(x)$ or $p_N(n)$	$F_X(x)$	$F_X^{-1}(p)$	X
beta	`beta.pdf(x, a, b, q, p)`	`beta.cdf(x, [a, b, q, p])`	`beta.inv(F, a, b, q, p)`	`beta.rnd(a, b, q, p, N, R, seed)`
binomial	`binomial.pmf(n, m, p)`	`binomial.cdf(n, m, p)`	--	`binomial.rnd(m, p, N, R, seed)`
chi2	`chi2.pdf(x, k)`	`chi2.cdf(x, k)`	`chi2.inv(p, k)`	`chi2.rnd(k, N, R, seed)`
exponential	`exponential.pdf(x, meanX)`	`exponential.cdf(x, meanX)`	`exponential.inv(P, meanX)`	`exponential.rnd(meanX, N, R, seed)`
extreme_value_I	`extreme_value_I.pdf(x, meanX, covnX)`	`extreme_value_I.cdf(x, [meanX, covnX])`	`extreme_value_I.inv(p, meanX, covnX)`	`extreme_value_I.rnd(meanX, covnX, N, R, seed)`
extreme_value_II	`extreme_value_II.pdf(x, m, s, k)`	`extreme_value_II.cdf(x, [m, s, k])`	`extreme_value_II.inv(P, m, s, k)`	`extreme_value_II.rnd(m, s, k, N, R, seed)`
gamma	`gamma.pdf(x, meanX, covnX)`	`gamma.cdf(x, [meanX, covnX])`	`gamma.inv(P, meanX, covnX)`	`gamma.rnd(meanX, covnX, N, R, seed)`
gev	`gev.pdf(x, m, s, k)`	`gev.cdf(x, [m, s, k])`	`gev.inv(p, m, s, k)`	`gev.rnd(m, s, k, N, R, seed)`
laplace	`laplace.pdf(x, meanX, sdvnX)`	`laplace.cdf(x, [meanX, sdvnX])`	`laplace.inv(P, meanX, sdvnX)`	`laplace.rnd(meanX, sdvnX, N, R, seed)`
lognormal	`lognormal.pdf(x, mednX, covnX)`	`lognormal.cdf(x, [mednX, covnX])`	`lognormal.inv(P, mednX, covnX)`	`lognormal.rnd(mednX, covnX, N, R, seed)`
normal	`normal.pdf(x, meanX, sdvnX)`	`normal.cdf(x, [meanX, sdvnX])`	`normal.inv(p, meanX, sdvnX)`	`normal.rnd(meanX, sdvnX, N, R, seed)`
poisson	`poisson.pmf(k, t, T)`	`poisson.cdf(k, t, T)`	--	`poisson.rnd(t, T, N, R, seed)`
quadratic	`quadratic.pdf(x, a, b)`	`quadratic.cdf(x, [a, b])`	`quadratic.inv(u, a, b)`	`quadratic.rnd(a, b, N, R, seed)`
rayleigh	`rayleigh.pdf(x, meanX)`	`rayleigh.cdf(x, meanX)`	`rayleigh.inv(P, meanX)`	`rayleigh.rnd(meanX, N, R, seed)`
students_t	`students_t.pdf(t, k)`	`students_t.cdf(t, k)`	`students_t.inv(p, k)`	`students_t.rnd(k, N, R, seed)`
triangular	`triangular.pdf(x, a, b, c)`	`triangular.cdf(x, [a, b, c])`	`triangular.inv(p, a, b, c)`	`triangular.rnd(a, b, c, N, R, seed)`
uniform	`uniform.pdf(x, a, b)`	`uniform.cdf(x, [a, b])`	`uniform.inv(F, a, b)`	`uniform.rnd(a, b, N, R, seed)`

Naming Conventions

Common Parameters:

- `x`, `t`, `k`, `n`: Evaluation points or event counts
- `N`: Number of samples (columns) to generate
- `R`: Correlation matrix (n × n), optional. If None, uncorrelated samples
- `seed`: Random seed for reproducibility, optional

Distribution-Specific Parameters:

Bounds & Location:

- `a`, `b`: Lower and upper bounds (uniform, triangular, quadratic, beta)
- `c`: Mode (triangular)
- `meanX`, `mednX`: Mean or median
- `m`: Location parameter (extreme value, GEV)

Scale & Shape:

- `sdvnX`: Standard deviation
- `covnX`: Coefficient of variation (std/mean)
- `s`: Scale parameter (extreme value, GEV)
- `k`: Shape parameter or degrees of freedom (chi2, students_t, GEV)
- `q`, `p`: Shape parameters (beta)

Event/Count Parameters:

- `m`: Number of attempts (binomial)
- `p`: Probability of success (binomial)
- `t`: Time duration (poisson)
- `T`: Mean return period (poisson)

Notes

1. **CDF signature pattern:**
   - Single-parameter distributions use direct parameter: `cdf(x, param)`
   - Multi-parameter distributions use params array: `cdf(x, [param1, param2, ...])`
   - This design facilitates curve fitting to empirical data
2. **Return shapes:**
   - PDF/PMF, CDF: Returns `(n, N)` array for n random variables
   - INV: Returns `(n, N)` or broadcast shape
   - RND: Returns `(n, N)` array, or `(N,)` if n=1 (flattened for single variable)
3. **Correlation:**
   - All `rnd()` functions support correlation matrix R via Gaussian copula method
   - Correlation applies to the underlying standard normal or uniform variates
4. **Discrete distributions:**
   - Binomial and Poisson have `pmf` (probability mass function) instead of `pdf`
   - No `inv` function for discrete distributions (not uniquely defined)

Example Usage

```
import numpy as np
from multivarious.rvs import normal, lognormal, beta

# Single uncorrelated variable
x = np.linspace(-3, 3, 100)
f = normal.pdf(x, meanX=0, sdvnX=1)

# Multiple correlated variables can be specified as simple "lists"
meanX = [1.0, 2.0, 3.0]  # mean values for three normal rv's
sdvnX = [0.5, 0.6, 0.7]  # standard devations for 3 normal rv's
R = [[1.0, 0.5, 0.3],    # orrelation matrix among three rv's
     [0.5, 1.0, 0.4],
     [0.3, 0.4, 1.0]])

# generate a sample 1000 values of three correlated normal rv's
X = normal.rnd(meanX, sdvnX, N=1000, R=R)  # Shape: (3, 1000)

# Compute the marginal CDFs for each of the three normal rv's
F = normal.cdf(x, [meanX, sdvnX])
```