The Normal Distribution - `normal`

The main commands

- `normal()` generates a set of random numbers from a standard Normal distribution.
- `normal(mu, sigma)` generates draws from a Normal distribution with mean μ and standard deviation σ.
- `normal(mu, sigma, (10,10))` generates a 10 by 10 array of draws from a Normal with mean μ and standard deviation σ.
- `normal(mu, sigma)` is equivalent to `mu + sigma * standard_normal()`.

The Poisson Distribution - `poisson`

- `poisson()` generates a set of random numbers from a Poisson distribution with $\lambda = 1$.
- `poisson(lambda)` generates a draw from a Poisson distribution with expectation λ.
- poisson(lambda, (10,10)) generates a 10 by 10 array of draws from a Poisson distribution with expectation λ.

`standard_t`

`standard_t(nu)` generates a set of random numbers from a Students t with shape parameter ν.
`standard_t(nu, (10,10))` generates a 10 by 10 array of draws from a Students t with shape parameter ν.

`uniform`

`uniform()` generates a uniform random variable on (0, 1).
`uniform(low, high)` generates a uniform on (l , h).
`uniform(low, high, (10,10))` generates a 10 by 10 array of uniforms on (l , h).

Continuous Random Variables

SciPy contains a large number of functions for working with continuous random variables. Each function resides in its own class (e.g. norm for Normal or gamma for Gamma), and classes expose methods for random number generation, computing the PDF, CDF and inverse CDF, fitting parameters using MLE, and computing various moments. The methods are listed below, where dist is a generic placeholder for the distribution name in SciPy.

- `dist.rvs`
  Pseudo-randomnumbergeneration. Generically, rvs is called using dist.rvs(*args, loc=0, scale=1, size=size) where size is an n-element tuple containing the size of the array to be generated.
- `dist.pdf`
  Probability density function evaluation for an array of data (element-by-element). Generically, pdf is called using `dist.pdf(x, *args, loc=0, scale=1)` where x is an array that contains the values to use when evaluating PDF.

- `dist.cdf`
  Cumulative distribution function evaluation for an array of data (element-by-element). Generically, cdf is called using `dist.cdf(x, *args, loc=0, scale=1)` where `x` is an array that contains the values to use when evaluating CDF.
- `dist.ppf`
  Inverse CDF evaluation (also known as percent point function) for an array of values between 0 and 1. Generically, ppf is called using `dist.ppf(p, *args, loc=0, scale=1)` where p is an array with all elements between 0 and 1 that contains the values to use when evaluating inverse CDF.

- `dist.fit`
  Estimate shape, location, and scale parameters from data by maximum likelihood using an array of data.
  Generically, fit is called using `dist.fit(data, *args, floc=0, fscale=1)` where data is a data array used to estimate the parameters.
  floc forces the location to a particular value (e.g. floc=0). `fscale` similarly forces the scale to a particular value (e.g. `fscale=1`) .
  It is necessary to use floc and/or fscale when computing MLEs if the distribution does not have a location and/or scale.
  For example, the gamma distribution is defined using 2 parameters, often referred to as shape and scale.
  In order to useMLto estimate parameters from a gamma, floc=0 must be used.

- `dist.median`
  Returns the median of the distribution. Generically, median is called using dist.median(*args, loc=0, scale=1).
- `dist.mean`
  Returns the mean of the distribution. Generically, mean is called using dist.mean(*args, loc=0, scale=1).
- `dist.moment`
  nth non-centralmomentevaluation of the distribution. Generically, moment is called using dist.moment(r, *args, loc=0, scale=1) where r is the order of the moment to compute.
- `dist.var`
  Returns the variance of the distribution. Generically, var is called using `dist.var(*args, loc=0, scale=1)`.
- `dist.std`
  Returns the standard deviation of the distribution. Generically, std is called using dist.std(*args, loc=0, scale=1).

Example

The gamma distribution is used as an example.
The gamma distribution takes 1 shape parameter a (a is the only element of *args), which is set to 2 in all examples.

```
>>> import scipy.stats as stats
>>> gamma = stats.gamma

>>> gamma.mean(2), gamma.median(2)
>>> gamma.std(2), gamma.var(2)
(2.0, 1.6783469900166608, 1.4142135623730951, 2.0)

>>> gamma.moment(2,2) gamma.
moment(1,2)**2 # Variance
```

```
>>> gamma.cdf(5, 2), gamma.pdf(5, 2)
(0.95957231800548726, 0.033689734995427337)

>>> gamma.ppf(.95957231800548726, 2)
5.0000000000000018

>>> log(gamma.pdf(5, 2)) gamma.
logpdf(5, 2)
0.0
```

```
>>> gamma.rvs(2, size=(2,2))
array([[ 1.83072394, 2.61422551],
       [ 1.31966169, 2.34600179]])

>>> gamma.fit(gamma.rvs(2, size=(1000)), floc = 0)
    # a, 0, shape
(2.209958533078413, 0, 0.8918726284546031З)
```