0.1 Random Number Generation with NumPy

NumPy random number generators are all stored in the module numpy.random. These can be imported with using import numpy as np and then calling np.random.rand, for example, or by importing import numpy.random as rnd andusing rnd.rand.1

0.1.1 rand, random_sample

randandrandom_sample are uniform randomnumber generators whichare identicalexceptthat rand takes a variable number of integer inputs – one for each dimension – while random_sample takes a n-element tuple.

random_sample is the preferred NumPy function, and rand is a convenience function primarily for MATLABusers.

```
>>> x = rand(3,4,5) >>> y = random_sample((3,4,5))
```

0.1.2 shuffle

shuffle randomlyreorderstheelementsofanarrayinplace.

```
>>> x = arange(10) >>> shuffle(x) >>> x array([4, 6, 3, 7, 9, 0, 2, 1, 8, 5])
```

0.1.3 permutation

permutation returns randomly reordered elements of an array as a copy while not directly changing the input.

```
>>> x = arange(10) >>> permutation(x) array([2, 5, 3, 0, 6, 1, 9, 8, 4, 7])
>>> x array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

0.2 Select Random Number Generators

All takebetween 0 and 2 required inputs which are parameters of the distribution, plus a tuple containing the size of the output. Allrandomnumbergeneratorsareinthemodule numpy.random.

0.2.1 median

median computed the median value in an array. An optional second argument provides the axis to use (default is to use entire array).

```
>>> x= randn(4,5)
>>> x
array([[0.74448693,
0.63673031,
0.40608815,
0.40529852, 0.93803737],
[ 0.77746525, 0.33487689, 0.78147524, 0.5050722
, 0.58048329],
[0.51451403,
0.79600763,
0.92590814, 0.53996231,
0.24834136],
[0.83610656,
0.29678017, 0.66112691,
0.10792584, 1.23180865]])
>>> median(x)
0.45558017286810903
>>> median(x, 0)
array([0.62950048,
0.16997507,
0.18769355, 0.19857318,
0.59318936])
```

Note that when an array or axis dimension contains an even number of elements (n), median returns the average of the 2 inner elements.

0.2.2 std

std computes the standard deviation of an array. An optional second argument provides the axis to use (default is to use entire array). std can be used either as a function or as a method on an array.

0.2.3 var

var computes the variance of an array. An optional second argument provides the axis to

....

0.2.4 corrcoef

corrcoef(x) computes the correlation between the rows of a 2-dimensional array x . corrcoef(x, y) computes the correlation between two 1- dimensional vectors. An optional keyword argument rowvar can be used to compute the correlation between the columns of the input – this is corrcoef(x, rowvar=False) and corrcoef(x.T) are identical

0.2.5 cov

cov(x) computes the covariance of an array x . cov(x,y) computes the covariance between two 1-dimensional vectors. An optional keyword argument rowvar can be used to compute the covariance between the columns of the input – this is cov(x, rowvar=False) and cov(x.T) are identical. histogram histogram can be used to compute the histogram (empirical frequency, using k bins) of a set of data. An optional second argument provides the number of bins. If omitted, k =10 bins are used. histogram returns two outputs, the first with a k-element vector containing the number of observations in each bin, and the second with the k + 1 endpoints of the k bins.

0.3 normaltest

normaltest tests for normality in an array of data. An optional second argument provides the axis to use (default is to use entire array). Returns the test statistic and the p-value of the test. This test is a small sample modified version of the Jarque-Bera test statistic.

0.4 kstest

kstest implements the Kolmogorov-Smirnov test. Requires two inputs, the data to use in the test and the distribution, which can be a string or a frozen random variable object. If the distribution is provided as a string, then any required shape parameters are passed in the third argument using a tuple containing these parameters, in order.

```
>>> x = randn(100)
>>> kstest = stats.kstest
>>> stat, pval = kstest(x, 'norm')
>>> stat
0.11526423481470172
>>> pval
0.12963296757465059
>>> ncdf = stats.norm().cdf # No () on cdf to get the function
>>> kstest(x, ncdf)
(0.11526423481470172, 0.12963296757465059)
>>> x = gamma.rvs(2, size = 100)
>>> kstest(x, 'gamma', (2,)) # (2,) contains the shape parameter
(0.079237623453142447, 0.54096739528138205)
>>> gcdf = gamma(2).cdf
>>> kstest(x, gcdf)
(0.079237623453142447, 0.54096739528138205)
```

0.4.1 ks_2samp

ks_2samp implements a 2-sample version of the Kolmogorov-Smirnov test. It is called ks_2samp(x,y) where both inputs are 1-dimensonal arrays, and returns the test statistic and p-value for the null that the distribution of x is the same as that of y . shapiro shapiro implements the Shapiro-Wilk test for normality on a 1-dimensional array of data. It returns the test statistic and p-value for the null of normality.

1 Statsmodels

Statsmodels is a Python module that allows users to explore data, estimate statistical models, and perform statistical tests. An extensive list of descriptive statistics, statistical tests, plotting functions, and result statistics are available for different types of data and each estimator. Researchers across fields may find that statsmodels fully meets their needs for statistical computing and data analysis in Python. Features include:

- Linear regression models
- Generalized linear models
- Discrete choice models
- Robust linear models
- Many models and functions for time series analysis
- Nonparametric estimators
- A collection of datasets for examples
- A wide range of statistical tests
- Input-output tools for producing tables in a number of formats (Text, LaTex, HTML) and for reading Stata files into NumPy and Pandas.
- Plotting functions
- Extensive unit tests to ensure correctness of results
- Many more models and extensions in development