Rough work

Rather than giving a summary of each individual function I would give a general explanation of what the library does and reference the various functions in that explanation.

Note that the numpy documentation already lists each function, what it does and what inputs and outputs it has. There's no point in just re-writing the documentation in your notebook. It would be better to write an illuminating notebook about what the library does overall.

Even though there are a lot of functions in the package, they all are actually variations on the basic notion of drawing values from probability distributions. For instance, the functions in "Simple random data" are just convenience functions for using the "Distributions" functions. There's not a lot to them and they're all quite similar.

Think of your notebook as giving the flavour of what the package is about, using plots, words, and code snippets. There's no point in re-writing the numpy documentation out.

Does that help?

Ian

1. Explain the overall purpose of the package
2. Explain the use of “Simple random data” and “Permutations” function
3. Explain the use and purpose of at least five “Distributions” functions
4. Explain the use of seeds in generating pseudorandom numbers

Requirements use git for version control

Use git hub and have a public repo

Have lots of commits at least tens. Each commit will be a small bit of work.

Must write it yourself

Minimum standard – includes

Need a README, gitignore and ipnb files. The ipnb should be the main body of work.

References

Clearly organised with concise explanations of datasets.

Interesting and well researched.

Use all functionality of jupyter including images, links, code and plots.

Use python libraries

Plan for numpy.random

**Why are random numbers needed?**

The ability to generate random numbers has many useful applications. Say a researcher is interested in the effectiveness of a new ulcer drug. The population they are interested in is every person with ulcers. It would impractically costly and time-consuming contact everyone with ulcers, give them the drug and then compare before and after reports. Indeed, the population of people with ulcers as a total population would be impossible to judge as many people may have ulcers without ever seeking treatment. The researcher needs to take a sample from this population. This sample should be big enough and diverse enough so that it fairly represents the wider population. The sample data is used to make generalisations about its population. For example, the drug’s effectiveness in a sample containing only men with ulcers aged seventy or over is unlikely to represent its effectiveness on women half that age. Sampling involves taking a sufficiently sized subset of a given population that accurately reflects the phenomena under investigation so that information from the sample can be used to infer things from the overall population. It is vital then that the sample shares the same characteristics as its target population.

**Problems with picking samples:**

Picking samples that represent a population is prone to biases. For example, if asked to pick a number at random between 1 and 10 there will be far more 3's and 7's than would be expected if the choice was truly random [ref](<https://micro.magnet.fsu.edu/creatures/pages/random.html>). One way to reduce sampling bias is to use random sampling. Random sampling means that every item in the target population has an equal chance of being selected. In the example above, the target population would be the numbers between 1 and 10 inclusive and random sampling means that each number in this interval would have an equal probability of being selected. If random sampling is not used then certain elements of the population are favoured over others which skews the results and limits its generalising ability. In reality, it is often difficult to define the target population and its sample precisely so that only the phenomena in question is investigated. For example in 1936 there was a phone poll conducted by The Literary Digest which predicted that the Republican Alfred Landon would win the election 57% to 43% from the Democrat F.D.Roosevelt. However, the actual results from the election was 62% in favour of FDR. The problem was sample bias. Only people with phones were polled and they were more likely to vote Republican in depression era America. People with phones were also more likely to come from a more affluent background and were not as likely to be moved by FDR’s new deal rhetoric. [Ref]( <https://www.math.upenn.edu/~deturck/m170/wk4/lecture/case1.html>). Random sampling does not remove sample bias but it does limit one aspect of it, biases introduced by favouring subsets of the target population rather than then every element being as likely to be selected as every other one.

**Algorithms and random numbers**

Computing algorithms are deterministic. They cannot generate true randomness without using some external piece of information. For example the DNS service Cloudflare uses its lava lamps in its head office to help generate the randomness needed for its cryptography [ref]( <https://blog.cloudflare.com/randomness-101-lavarand-in-production/>). Many areas of science and computing do not need true randomness. In fact, some areas require predictable ‘randomness’ so that model parameters can be tested with the same random ‘noise’. Computer algorithms use pseudo random numbers (Idris, 2015). To all appearances these look random but if a key piece of information is known, the whole random sequence can be predicted. There are several pseudo random number generator (PRNG) algorithms. One of the most widely used PRNG is the Mersenne Twister. A Mersenne number is one less than a power of 2. The twister aspect refers to its period length being a Mersenne prime which is ‘twisted’ by various transformations when random numbers are generated. A commonly used Mersenne twister uses 19937 as its power. This algorithm takes a ‘seed’ value as a starting point. This is initialised into a state and transformed via reversable and non-reversable transformations in order to generate the pseudo random numbers PRN. {REf}(https://www.cryptologie.net/article/331/how-does-the-mersennes-twister-work/). The Mersenne Twister is not secure enough for cryptology however it a useful and widely used all purpose PRNG [Ref]( https://en.wikipedia.org/wiki/Mersenne\_Twister).

**Random numbers in Python**

Python has an in built random package. NumPy extends this by adding extra functionality and methods in its numpy.random package. Both Python and Numpy.random use the Mersenne twister 19937 algorithm to generate PRN.

**Why use numpy for generating random numbers**

Python has a random method that generates a random float number uniformly from the interval 0 inclusive to 1 exclusive [Ref]( https://docs.python.org/3/library/random.html). Python.random has many of the functions and methods contained in numpy.random so why use numpy.random? The main reason numpy.random is used over python’s random.random package is that numpy is designed to work with n dimensional arrays. Numpy uses less memory and is faster than lists used in the default python. Numpy is specialised for scientific operations and has more advanced mathematical functionality than python. It is especially useful for manipulating numerical data that can be arranged in matrices [] (https://metaspace.blog/programming/python/python-numpy-basics/). PRN’s can be arranged in arrays, a speciality in NumPy. Numpy.random and python’s default random.random have similar functions and methods but numpy.random has some extra probability distributions common in scientific research and some extra simple random data convenience functions. Neither are suitable for cryptography purposes. [](<https://stackoverflow.com/questions/7029993/differences-between-numpy-random-and-random-random-in-python>).

The functions and methods offered in numpy.random are

* simple random data
* permutations
* distributions
* random generator

Simple random data section consist of several functions that generate simple random data.Inputs can be size or range. Permutation functions randomly shuffle or permutes a given sequence. Distribution functions allow specific population distributions or ranges to be sampled. This may require statistical measures such as mean and standard deviation. Numpy.random can generate samples from continuous and descrite distributins (Idris, 2015). Lastly, the random generator section contains functions that allow seeds to be specified (Mehta, 2015). This is useful in cases where the exact same sequence of random data is required. Each of these sections will be explored.

**What does Simple random data do?**

**What does permutations do?**

**What are distributions and why do we need them.**

**What are seeds and how are they involved in generating pseudorandom numbers**.