User’s Guide to Running the Draft NIST SP 800-90B Entropy Estimation Suite

19 April 2016

K. McKay

This is a brief introduction on how to run the Python command-line programs (hosted on GitHub at <https://github.com/usnistgov/SP800-90B_EntropyAssessment>) that implement the statistical entropy estimation methods found in Section 6 of the [Second Draft NIST Special Publication (SP) 800-90B (January 2016)](http://csrc.nist.gov/publications/drafts/800-90/sp800-90b_second_draft.pdf). It is not a description or explanation of the methods themselves. Please refer to the draft SP for definitions and descriptions of the methods and their rationales.

# Disclaimer

This software was developed by employees of the National Institute of Standards and Technology (NIST), an agency of the Federal Government. Pursuant to title 15 United States Code Section 105, works of NIST employees are not subject to copyright protection in the United States and are considered to be in the public domain. As a result, a formal license is not needed to use the software.

This software is provided by NIST as a service and is expressly provided "AS IS". NIST MAKES NO WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT AND DATA ACCURACY. NIST does not warrant or make any representations regarding the use of the software or the results thereof including, but not limited to, the correctness, accuracy, reliability or usefulness of the software.

Permission to use this software is contingent upon your acceptance of the terms of this agreement.

The identification of any commercial product or trade name does not imply endorsement or recommendation by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

# Requirements

The code should run on any operating system with 64-bit Python 2.7 or Python 3.

Note that this tool does not come with a Python installation. If you do not already have Python installed on your system, go to [https://www.python.org](https://www.python.org/) and select “Download.” No additional modules or packages are required to run the code. However, some routines will run faster if you have the **numpy** package installed. You can get **numpy** at <http://www.scipy.org>. If you are running a Windows OS, you can also find it here: <http://www.lfd.uci.edu/~gohlke/pythonlibs>. Alternatively, you can download the entire **scipy-stack**, which includes **numpy**.

# Python Files

SP 800-90B breaks the process into two paths: an independent and identically distributed (IID) path and a non-IID path. The python files for each path are listed below.

### Both paths:

* ***util90b.py***
  + Contains utility functions, such as command line parser and loading data file
* ***restart.py***
  + Main file for the sanity checks on the restart dataset
* ***mostCommonValue.py***
  + Contains the most common value method for restart tests and the most common value estimate for IID and non-IID paths

### IID path:

* ***iid\_main.py***
  + Contains main routine to give the independent and identically distributed (IID) entropy estimate, if IID assumption holds
  + Run permutation tests to determine if IID
  + Run chi-square independence and goodness of fit tests to determine if IID
  + Run longest repeated substring test
  + Estimate min entropy if passes above tests
* ***permutation\_tests.py***
  + Contains tests to determine if dataset is IID
* ***chi\_square\_tests.py***
  + Contains the chi square independence and goodness of fit for binary and non-binary data
* ***LRS.py***
  + Contains the length of the longest repeated substring (LRS) test

### Non-IID path:

* ***noniid\_main.py***
  + Contains main routine to compute the non-IID entropy estimate
  + Runs ten methods to estimate min-entropy
  + Assessed min-entropy is the lowest of the ten results
* ***noniid\_collision.py***
  + Contains the collision estimate method
* ***markov.py***
  + Contains the Markov estimate method
  + Only up to 6 bits per symbol are used for the Markov test
* ***maurer.py***
  + Contains the compression estimate method
* ***tuple.py***
  + Contains the *t*-tuple estimate method
* ***LRS.py***
  + Contains the length of the longest repeated substring (LRS) test
* ***SP90Bv2\_predictors.py***
  + Contains the prediction estimates:
    - Multi most common in window estimate
    - Lag prediction estimate
    - multiMMC prediction estimate
    - LZ78Y prediction estimate

# Dataset

The code package expects the dataset to be a binary file where the symbols are stored as bytes. Each byte may only belong to one symbol. For example, an 8-bit symbol would be represented by all 8 bits of a byte, whereas a binary value would take up only the least significant bit of a byte (i.e., multiple bits cannot be packed into a byte). The number of bits per symbol is supplied to the code package via command line argument.

## Restart Dataset

The code package expects the restart dataset to be a concatenation (denoted by ||) of 1000 sequences of 1000 samples. If three sequences generated after three consecutive restarts were *s1*, *s2*, and *s3*, respectively, the restart dataset would be *s1* || *s2* || *s3*, in the format described above. In other words, this is the row dataset described in Section 3.1.4.1 of draft SP 800-90B. The code package constructs the column dataset from the row dataset.

## Sample Dataset Files:

This code package contains three dataset files generated with TrueRand that should pass the IID tests.

* 1000000 data samples
  + 1 bit per sample (***truerand\_1bit.bin***)
  + 4 bits per sample (***truerand\_4bit.bin***)
  + 8 bits per sample (***truerand\_8bit.bin***)

There is also one file containing binary digits of pi:

* ***data.pi.bin*** (1165666 bytes)

# Documentation

This user guide:

* ***user\_guide.pdf***

# Running the Code

## Initial Estimate for Non-IID Path

To obtain an entropy estimate using the non-IID path, the file ***noniid\_main.py*** should be executed. The help message for the non-IID tests is shown in the following example.

|  |
| --- |
| $ python noniid\_main.py -h  usage: noniid\_main.py [-h] [-u use\_bits] [-v] datafile bits\_per\_symbol  Run the Draft NIST SP 800-90B (January 2016) non-IID Tests  positional arguments:  datafile dataset on which to run tests  bits\_per\_symbol number of bits used to represent sample output values  optional arguments:  -h, --help show this help message and exit  -u use\_bits, --usebits use\_bits  use only the N lowest order bits per sample  -v, --verbose verbose mode: show detailed test results |

To run the code for the non-IID path, two arguments are required: the binary datafile and the number of bits per symbol. The datafile is a binary file containing output from an entropy source, and bits\_per\_symbol tells the program how many bits to use to construct each symbol. The program supports bits\_per\_symbol values from 1 to 8. While SP 800-90B can be applied to sources with greater symbols sizes, this program assumes that the reduction operation in Section 6.4 has been applied and the max symbol size is a byte.

There are two flags that may be set as well. Setting the verbose flag (-v) enables the program to print useful information about the progress of the computations and the results of individual estimation methods. The use bits flag (-u) and accompanying value tell the program to only test the use\_bits least significant bits of each symbol for estimation. This can be useful if all of the entropy is in lower order bits.

The following example shows the output for the initial non-IID entropy estimate, with the verbose flag set. The data is stored in bytes.

|  |
| --- |
| $ python noniid\_main.py -v truerand\_8bit.bin 8  reading 1000000 bytes of data  Read in file truerand\_8bit.bin, 1000000 bytes long.  Dataset: 1000000 8-bit symbols, 256 symbols in alphabet.  Output symbol values: min = 0, max = 255  Running entropic statistic estimates:  - Most Common Value Estimate: p(max) = 0.00428909, min-entropy = 7.86511  - Collision Estimate: p(max) = 0.0127255, min-entropy = 6.29613  - Markov Estimate (map 6 bits): p(max) = 1.13787e-223, min-entropy = 5.78597  - Compression Estimate: p(max) = 0.00872433, min-entropy = 6.84074  - t-Tuple Estimate: p(max) = 0.004124, min-entropy = 7.92174  - LRS Estimate: p(max) = 0.00391357, min-entropy = 7.9973  Running predictor estimates:  Computing MultiMCW Prediction Estimate: 100 percent complete  Pglobal: 0.003937  Plocal: 0.002136  MultiMCW Prediction Estimate: p(max) = 0.0039373, min-entropy = 7.98858  Computing Lag Prediction Estimate: 100 percent complete  Pglobal: 0.004073  Plocal: 0.002136  Lag Prediction Estimate: p(max) = 0.00407281, min-entropy = 7.93976  Computing MultiMMC Prediction Estimate: 100 percent complete  Pglobal: 0.004110  Plocal: 0.002136  MultiMMC Prediction Estimate: p(max) = 0.00410955, min-entropy = 7.92681  Computing LZ78Y Prediction Estimate: 100 percent complete  Pglobal: 0.004110  Plocal: 0.002136  LZ78Y Prediction Estimate: p(max) = 0.00410961, min-entropy = 7.92678  -----------------------  min-entropy = 5.78597  Don't forget to run the sanity check on a restart dataset using H\_I = 5.78597 |

The output for the same computations without the verbose flag is:

|  |
| --- |
| $ python noniid\_main.py truerand\_8bit.bin 8  reading 1000000 bytes of data  -----------------------  min-entropy = 5.78597  Don't forget to run the sanity check on a restart dataset using H\_I = 5.78597 |

The resulting H\_I (in this example, 5.78597) is the initial entropy estimate. It is used as an input to the restart test, described below.

If the entropy were all in the lower-order bits, then it would be desirable to use the –u flag. The following example shows computations on the same data file, but using only the four low-order bits of each byte.

|  |
| --- |
| $ python noniid\_main.py -v -u 4 truerand\_8bit.bin 8  reading 1000000 bytes of data  Read in file truerand\_8bit.bin, 1000000 bytes long.  Dataset: 1000000 8-bit symbols, 256 symbols in alphabet.  Output symbol values: min = 0, max = 255  \* Using only low 4 bits out of 8. 16 symbols in reduced alphabet.  \* Using output symbol values: min = 0, max = 15  Running entropic statistic estimates:  - Most Common Value Estimate: p(max) = 0.0635666, min-entropy = 3.97559  - Collision Estimate: p(max) = 0.0852737, min-entropy = 3.55175  - Markov Estimate: p(max) = 3.53812e-152, min-entropy = 3.93055  - Compression Estimate: p(max) = 0.0793228, min-entropy = 3.65612  - t-Tuple Estimate: p(max) = 0.0774597, min-entropy = 3.69041  - LRS Estimate: p(max) = 0.0676245, min-entropy = 3.88631  Running predictor estimates:  Computing MultiMCW Prediction Estimate: 100 percent complete  Pglobal: 0.062795  Plocal: 0.025391  MultiMCW Prediction Estimate: p(max) = 0.062795, min-entropy = 3.99321  Computing Lag Prediction Estimate: 100 percent complete  Pglobal: 0.063075  Plocal: 0.025391  Lag Prediction Estimate: p(max) = 0.0630754, min-entropy = 3.98678  Computing MultiMMC Prediction Estimate: 100 percent complete  Pglobal: 0.062967  Plocal: 0.046875  MultiMMC Prediction Estimate: p(max) = 0.0629669, min-entropy = 3.98926  Computing LZ78Y Prediction Estimate: 100 percent complete  Pglobal: 0.063162  Plocal: 0.046875  LZ78Y Prediction Estimate: p(max) = 0.0631618, min-entropy = 3.9848  -----------------------  min-entropy = 3.55175  Don't forget to run the sanity check on a restart dataset using H\_I = 3.55175 |

After the non-IID estimate is returned, the sanity checks on the restart dataset must be applied as described below.

## Initial Estimate for IID Path

To test whether a dataset is IID and obtain an entropy estimate for that dataset, the file ***iid\_main.py*** should be executed. The help message for the IID tests is shown is as follows:

|  |
| --- |
| $ python iid\_main.py -h  usage: iid\_main.py [-h] [-v] datafile bits\_per\_symbol  Run the Draft NIST SP 800-90B (January 2016) IID Tests  positional arguments:  datafile dataset on which to run tests  bits\_per\_symbol number of bits used to represent sample output values  optional arguments:  -h, --help show this help message and exit  -v, --verbose verbose mode: show detailed test results |

To run the code for the IID path, two arguments are required: the binary datafile and the number of bits per symbol. The following examples uses the datafile truerand\_8bit.bin, which is provided with this package, and the bits\_per\_symbol is 8. If the verbose flag is set, information about the dataset is provided. This information includes the number of bytes, the number of bits per symbol, the number of unique symbols observed, and the minimum and maximum values.

|  |
| --- |
| $ python iid\_main.py -v truerand\_8bit.bin 8  reading 1000000 bytes of data  Read in file truerand\_8bit.bin, 1000000 bytes long.  Dataset: 1000000 8-bit symbols, 256 symbols in alphabet.  Output symbol values: min = 0, max = 255 |

The permutation tests take hours to compute. Unlike the code that was released with the 2012 draft, this version of the 90B code package does not allow the user to reduce the number of permutations performed. In addition, the permutation tests apply 10000 permutations on the full sequence, rather than 1000 permutations on ten data subsets as was done in the 2012 draft. While the permutation tests are running, the status will be displayed when the verbose flag is set. This can be seen in the following incomplete execution of the IID process.

|  |
| --- |
| $ python iid\_main.py -v truerand\_8bit.bin 8  reading 1000000 bytes of data  Read in file truerand\_8bit.bin, 1000000 bytes long.  Dataset: 1000000 8-bit symbols, 256 symbols in alphabet.  Output symbol values: min = 0, max = 255  Calculating statistics on original sequence  Calculating statistics on permuted sequences  permutation tests: 31.10 percent complete |

If the dataset passes all of the permutation tests, as is the case for truerand\_8bit.bin, then the program output indicates this and moves on to the Chi-square tests. If those are passed, the program output indicates this and applies the length of the longest repeated substring test. If that passes, then the program outputs “IID = True” and then provides an entropy estimate. If any of these tests fail, the program outputs “IID = False” and exits.

|  |
| --- |
| $ python iid\_main.py -v truerand\_1bit.bin 1  reading 1000000 bytes of data  Read in file truerand\_1bit.bin, 1000000 bytes long.  Dataset: 1000000 1-bit symbols, 2 symbols in alphabet.  Output symbol values: min = 0, max = 1  Calculating statistics on original sequence  Calculating statistics on permuted sequences  permutation tests: 99.99 percent complete  statistic C[i][0] C[i][1]  excursion 5486 0  numDirectionalRuns 4272 62  lenDirectionalRuns 1175 2368  numIncreasesDecreases 8992 44  numRunsMedian 8429 296  lenRunsMedian 1024 7  avgCollision 148 1  maxCollision 1307 366  periodicity(1) 7931 68  periodicity(2) 4035 78  periodicity(8) 3195 70  periodicity(16) 9532 26  periodicity(32) 263 17  covariance(1) 1706 1  covariance(2) 1883 2  covariance(8) 1285 2  covariance(16) 2831 1  covariance(32) 657 0  compression 7153 62  (\* denotes failed test)  \*\* Passed IID permutation tests  Chi square independence  score = 1949.69, degrees of freedom = 2047, cut-off = 2250.43  \*\* Passed chi-square independence test  Chi square goodness-of-fit  score = 2.56106, degrees of freedom = 9 cut-off = 27.877  \*\* Passed chi-square goodness-of-fit test  \*\* Passed chi square tests  LRS test  W: 36, Pr(E>=1): 1.0)  \*\* Passed LRS test  IID = True  min-entropy = 0.995043  Don't forget to run the sanity check on a restart dataset using H\_I = 0.995043 |

If the verbose flag is not set, the output shows only the final results. Specifically, whether IID is true or false, and if true, what the min-entropy estimate is.

After the IID estimate is returned, the sanity checks on the restart dataset must be applied as described below.

## Restart Tests

The main file for the restart tests is ***restart.py***, which requires two arguments and has an optional verbose flag. The first required argument is the row dataset, as defined in Section 3.1.4.1 of draft SP 800-90B. The program derives the column dataset from the row dataset, so ***restart.py*** only needs to be run once.

If the file truerand\_8bit.bin were a row dataset, the restart tests would be performed as follows (with verbose on).

|  |
| --- |
| $ python restart.py -v truerand\_8bit.bin 8 5.78597  reading 1000000 bytes of data  Read in file truerand\_8bit.bin, 1000000 bytes long.  Dataset: 1000000 8-bit symbols, 256 symbols in alphabet.  Output symbol values: min = 0, max = 255  Running sanity check on row dataset:  - F\_R: 16  Running sanity check on column dataset:  - F\_C: 15  alpha: 1.953125e-08  z: 5.61610279  U: 41.815068515  Passed the restart tests  \*\*\* Final entropy estimate: 5.78597 |

Suppose that the initial entropy estimate had been 7.9. Then the restart tests would fail, as shown in the following example.

|  |
| --- |
| $ python restart.py -v truerand\_8bit.bin 8 7.9  reading 1000000 bytes of data  Read in file truerand\_8bit.bin, 1000000 bytes long.  Dataset: 1000000 8-bit symbols, 256 symbols in alphabet.  Output symbol values: min = 0, max = 255  Running sanity check on row dataset:  - F\_R: 16  Running sanity check on column dataset:  - F\_C: 15  U: 15.653766  Failed the restart tests  \*\*\* Validation failed. No entropy estimate awarded. |