(DRAFT) User's Guide to Running the Draft NIST SP 800-90B Entropy Estimation Suite

17 March 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 SP 800-90B (January 2016). 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

Requirements

The code should run on any OS with 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 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 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

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
- ***mostCommonValue.py***
  - Contains the Most Common Value Estimate

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
- ***mostCommonValue.py***
  - Contains the most common value estimate method
- ***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.

- 1 000 000 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 is available in two formats:

- ***user_guide.docx***
- ***user_guide.pdf***

Running the Code

Initial Estimate for Non-IID Path

To follow the non-IID path, the first file that should be executed is ***noniid_main.py***. 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: 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:
MultiMCW: 100 percent
     Pglobal: 0.003937
     Plocal: 0.002136
- MultiMCW Prediction Estimate: p(max) = 0.0039373, min-entropy = 7.98858
Lag: 100 percent
     Pglobal: 0.004073
     Plocal: 0.002136
- Lag Prediction Estimate: p(max) = 0.00407281, min-entropy = 7.93976
MultiMMC: 100 percent
     Pglobal: 0.004110
     Plocal: 0.002136
- MultiMMC Prediction Estimate: p(max) = 0.00410955, min-entropy = 7.92681
LZ78Y:   100 percent
     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: 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) = 0.0793076, min-entropy = 3.6564
- t-Tuple Estimate: p(max) = 0.0774597, min-entropy = 3.69041
- LRS Estimate: p(max) = 0.0676245, min-entropy = 3.88631

Running predictor estimates:
MultiMCW: 100 percent
     Pglobal: 0.062795
     Plocal: 0.025391
- MultiMCW Prediction Estimate: p(max) = 0.062795, min-entropy = 3.99321
Lag: 100 percent
     Pglobal: 0.063075
     Plocal: 0.025391
- Lag Prediction Estimate: p(max) = 0.0630754, min-entropy = 3.98678
MultiMMC: 100 percent
     Pglobal: 0.062967
     Plocal: 0.046875
- MultiMMC Prediction Estimate: p(max) = 0.0629669, min-entropy = 3.98926
LZ78Y:    100 percent
     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 follow the IID path, the first file that should be executed is ***iid_main.py***. 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 of 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.
```