BitReps - University of Kent - QHub2

Jamie Pont, Calvin Brierley, Julio Hernandez-Castro

BitReps is a statistical test for identifying signs of non-randomness within the output of an RNG, developed by the University of Kent as part of the Quantum Communications Hub, Phase 2.0.

BitReps is available to clone from GitHub.

BitReps identifies repetition within RNG output at the level of bits, statistically quantifying the level of repetition observed compared to that of an expected distribution dervied from known "random" data.

Contents

How does it work?

For a set $\{1, ..., x\}$ sampled n times, the number of unique values expected in the output is given by:

$$x[1 - (1 - 1/x)^n]$$

Under any given blocksize b (in bits), the output of an RNG can be considered the sampling of the set $\{0, ..., 2^b\}$. Therefore, using the above formula, BitReps calculates the expected number of genuine repetitions for a given RNG output where $x = 2^b$ and $n = \text{number of blocks}$.

In practice, BitReps uses a bloom filter, a probabilistic data structure prone to false positives and therefore a number of additional "repetitions" are accounted for (which are in reality just false positives).

Requirements

- Ubuntu 20.04.4 LTS
    - Up-to-date installations of Windows and/or MacOS will work, although BitReps was developed on, and so far solely tested using, Ubuntu
- Python 3.8.10+
    - Older (yet still supported) versions of Python will likely work, although it is recommended to use the latest version of Python to avoid issues such as package incompatibilities

Installation - Initial Setup

Clone the BitReps repository

```
git clone [URL]
```

Set up a virtual environment

```
python3 -m venv env
```

Activate the virtual environment

Unix/MacOS: `source env/bin/activate`

Windows: `.\env\bin\activate`

Verify successful setup

`pip -V` (Update Pip if prompted)

(The virtual environment can be deactivated at any time using `deactivate`)

Installation - BitReps Setup

Install project dependencies

```
pip install -r requirements.txt
```

Start BitReps

```
python gui.py
```

Directory Structure

```
BitReps
|    .gitignore
|    README.md
|    README.pdf
|    requirements.txt
|    gui.py
|    main.py
|    processor.py
└--- input
|    |    urandom100M-2.bin
|
└--- model
     |    urandom100M-32-1e-05-False.json
```

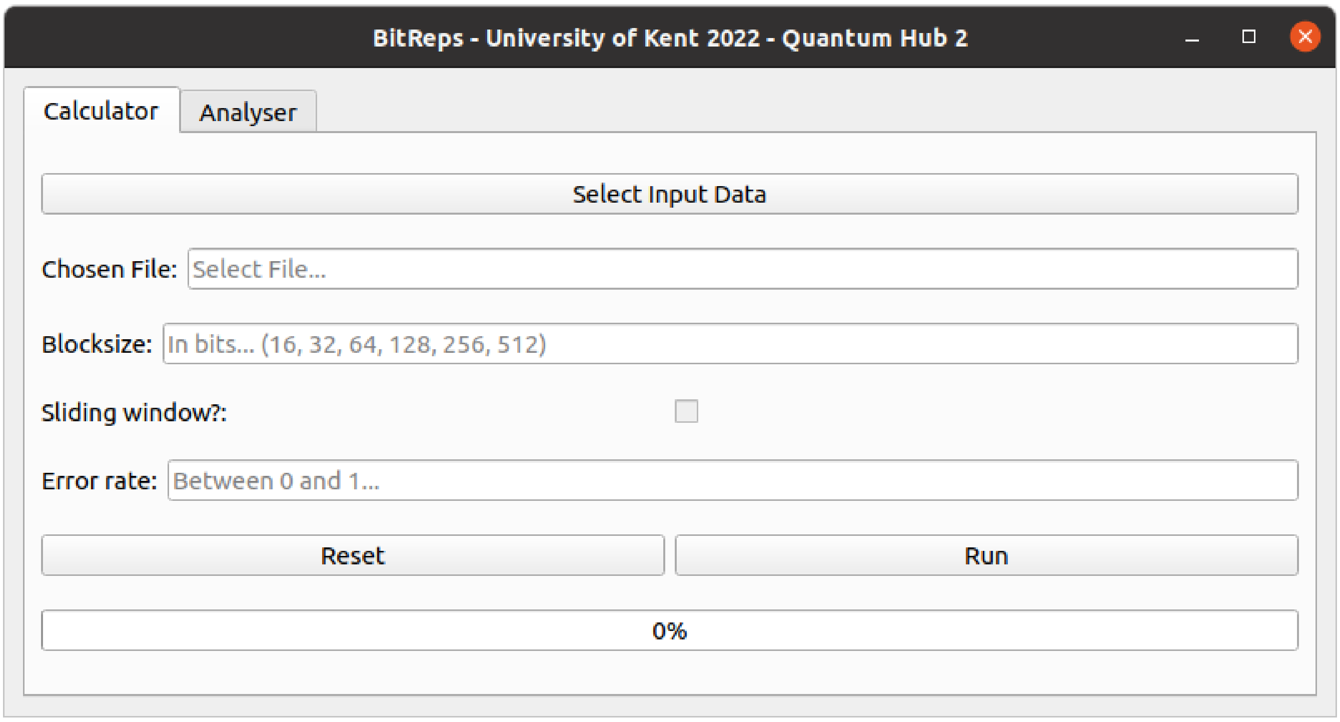
Basic Usage

In the following subsections, a brief overview of using BitReps is provided. When launching BitReps, the main GUI will be displayed from which all features can be accessed. BitReps has two primary functions, i.e. measuring the number and locations of repetitions in RNG output (performed by the Calculator) and performing a statistical analysis over these repetitions (performed by the Analyser).

In addition, the following directories will be created in the root directory to handle BitReps measurements and statistical analysis output:

```
BitReps
|    ... (as above) ...
└--- output
└--- results
```

Calculator


BitReps - University of Kent 2022 - Quantum Hub 2
Calculator
Analyser
Select Input Data
Chosen File: Select File...
Blocksize: In bits... (16, 32, 64, 128, 256, 512)
Sliding window?: □
Error rate: Between 0 and 1...
Reset
Run
0%

The **Select Input Data** button can be used to select RNG output over which to measure repetitions. After selecting a file, the following parameters should be set:

- Blocksize: The number of bits over which a repetition is measured (values supported: $8, 16, 32, 64, 128, 512$)
- Sliding Window: If selected, shift the block by one bit at a time, rather than by $\mathrm{blocksize}$ bits
- Error Rate: The maximum desired error rate of the underlying bloom filter

Note: A file of example RNG output is included in the `input` directory which can be used to experiment with the BitReps tool. The RNG used to generate this data was `/dev/urandom`, a PRNG found on Unix-like operating systems, and the file is named: `urandom100M-2.bin`.

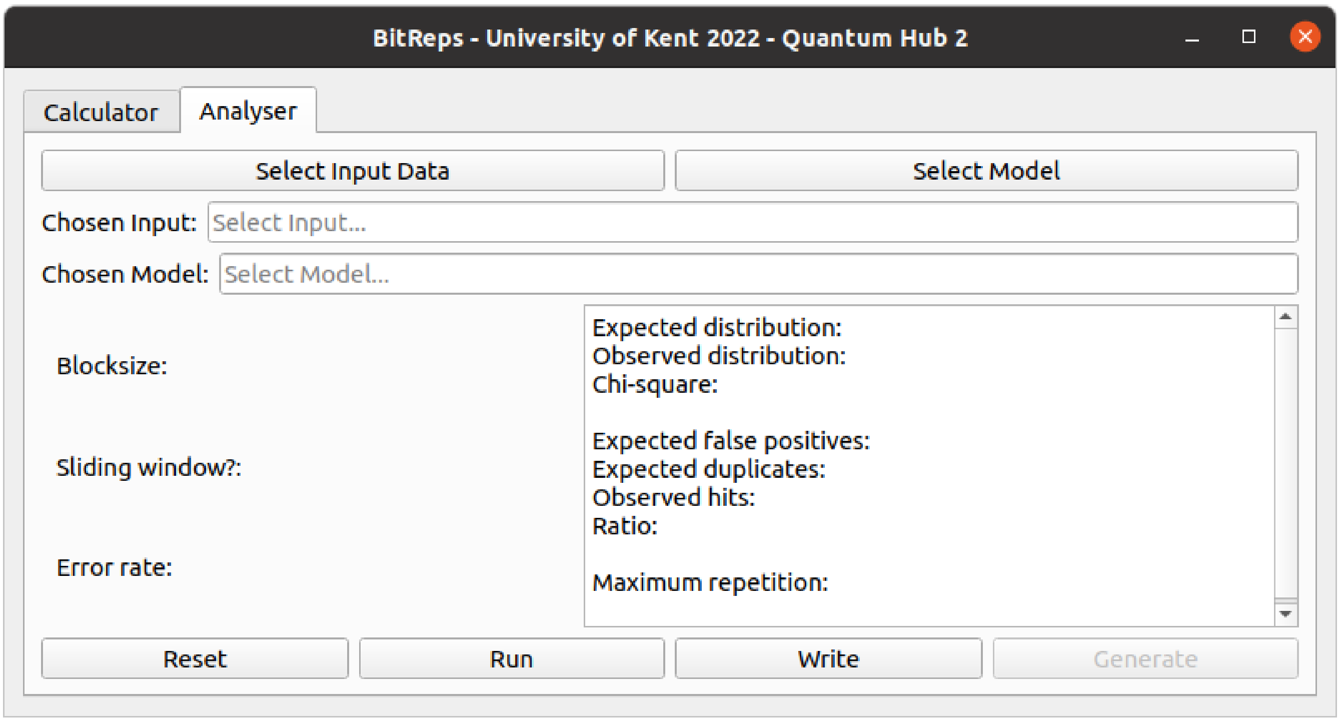
Once the parameters are set, the **Run** button should be used to conduct measurements. A progress bar (both on the GUI and command line) indicates progress.

Once measurements are complete, a file is written to the `output` directory using the following naming convention:

```
[filename of RNG data]-blocksize-errorrate-slidingwindow.json
```

At any point before Run is clicked, the **Reset** button can be used to reset all user input.

Analyser



The **Select Input Data** button can be used to select BitReps measurements over which to perform statistical analysis. The **Select Model** button can be used to select a baseline distribution against which the RNG repetitions are compared. Once these files are selected, the **Calculate** button can be used to perform automated analysis.

Note: A model file is included in the `model` directory which was measured against `/dev/urandom`. The model file is named: `urandom100M-32-1e-05-False.json`.

Output on the lefthand side of the screen represents metadata regarding BitReps analysis, and directly corresponds to the parameters entered on the Calculator tab.

Output on the righthand side of the screen is as follows:

- *Expected distribution:* The expected distribution used in the chi-square calculation
- *Observed distribution:* The observed distribution used in the chi-square calculation
- *Chi-square:* The chi-square value of the observed vs. expected distribution
- *Expected false positives:* The expected number of false positives given the size of the bloom filter
- *Expected duplicates:* The expected number of genuine repetitions given the blocksize and size of the RNG output
- *Observed hits:* The number of repetitions measured by BitReps

- *Ratio:* $\frac{\text{No. expected repetitions}}{\text{No. observed repetitions}}$
- *Maximum repetition:* The singlemost repeating "word" in the RNG output (as well as its binary representation)

The **Write** button can be used to write the information from the righthand side of the GUI to a file in the `results` directory using the following naming convention:

```
[filename of RNG data]-blocksize-errorrate-slidingwindow.txt
```

The **Generate** button (reserved for a future release) can be used to generate various graphical visualisations of the repetition within given RNG output with respect to the frequency and period of repetition.

The **Reset** button can be used to reset the metadata as well as the results obtained from automated analysis.

Note: Please ensure that statistical analyses are conducted over a model and input file which were built (using the Calculator tab) over RNG output of the same length.

For example, the included model file was built over 100MB of data and thus, if it is used, input files should also be built over 100MB of data. Whilst the analyses can be conducted over RNG output of differing lengths, the results will not be informative as of the current release.

Results Interpretation

After using the BitReps calculator, JSON output representing the repetitions within RNG output is saved to the ./output directory. You are then able to perform automated analysis and (in an upcoming release) generate graphs over this data by using the Analyser tab, as described above.

The below text boxes represent the results of two separate BitReps experiments. The first represents an analysis of data that can be considered random, and the second an analysis of data that can be considered non-random.

```
Random Data -

Expected distribution: [80141, 178]
Observed distribution: [79487, 164]
Chi-square: 6.44

Expected false positives: 20
Expected duplicates: 79837
Observed hits: 79651
Ratio: 1.0

Maximum repetition: 2 (11000011000010111011101000110100)
```

```
Non-Random Data -

Expected distribution: [80141, 178]
Observed distribution: [1742239, 581907]
Chi-square: 1935643441.75

Expected false positives: 20
Expected duplicates: 79837
Observed hits: 3177644
Ratio: 0.03

Maximum repetition: 3016 (00000000000000000000000000000000)
```

As can be seen above, the analyser output displays the exact distributions used for the chi-square calculations, and can be interpretted as follows. Using the obersved distribution for random data as an example, blocks which repeat once (i.e. are present in the RNG output twice) occur 79,487 times, and blocks which repeat twice (i.e. are present in the RNG output three times) occur 164 times. The index within the list dictates the number of times a repetition is observed, starting from one.

In the first set of results, it is clear that the observed distribution is similar to that of the expected distribution and thus a low chi-square value is achievied. This indication of randomness is corroborated by a ratio of 1.0, indicating an almost identical number of observed hits compared to expected hits (when mathematically modelling the expected number of bloom filter false positives and the expected number of genuine repetitions in the RNG output).

In the second set of results, the vastly differing distributions reflect a chi-square value many orders of magnitude larger than that of the first set of results. Similarly, the very low ratio of 0.03 indicates a vast difference between the number of observed and expected hits.

The first set of results measures that the maximally repeating block occurs twice in the output, whereas the maximally repeating block in the second set of results occurs 3016 times (and in this case is comprised entirely of zeros, perhaps indicating a faulty RNG device).

Upcoming Developments

BitReps is an ongoing project and many upcoming features are planned. These features include:

- Graphical visualisation
    - It was originally planned to include histograms representing the chi-square tests however the small number of bins resulted in graphs which lacked any real value. In an upcoming release, heatmaps will be added representing the level of repetition across the entire RNG output with respect to the chosen blocksize. This will help to identify patterns of repetition (such as their period and location) at a glance.
- Granular repetition
    - Currently, bitreps identifies repeating patterns across the entire RNG output. In a future release, we plan to run BitReps at a more granular level - i.e. split the RNG output into blocks of (say) 512 bits, and then identify much shorter repeating patterns within each block (for example strings of length 4 bits).
- Optimisation
    - BitReps development has focused on functionality - priority will be placed on optimisation such that runtimes are reduced (allowing for more experimentation of sliding window mode), JSON output is minimised to reduce the BitReps footprint, and BitReps processing is threaded to allow for statistical analysis in parallel with BitReps measurements.
- Quality of Life Improvements
    - Various QoL features are planned for BitReps to improve user experience. For example, in a future release, multiple RNG output files can be selected for BitReps measurements rather than having to select them individually, along with many more QoL and cosmetic improvements.