The Journal of Open Source Software

DiscreteEntropy.jl: Entropy Estimation of Discrete Random Variables with Julia

David A. Kelly [1¶] **and Ilaria Pia La Torre** [2]

1 King's College London, UK **2** University College London, UK ¶ Corresponding author

DOI: 10.21105/joss.07334

Software

- Review
- Repository
- Archive

Editor: Nikoleta Glynatsi

Reviewers:

- @niyiyu
- @nluetts

Submitted: 20 September 2024
Published: 12 November 2024

Summary

`DiscreteEntropy.jl` is a Julia package to facilitate entropy estimation of discrete random variables. The entropy of a random variable, `X`, is the average amount of surprise associated with the different outcomes of that variable. When `X` is known completely, calculating the entropy is easy. It is given by

$$H(X) = -\sum_{x \in X} p(x) \log(p(x))$$

However, it is a very hard problem when knowledge of the distribution is incomplete. It is well known that the `MaximumLikelihood` (or Plugin) estimator underestimates the true entropy on average (Basharin, 1959). This difficulty has led to a large number of improved estimators. Contreras Rodríguez et al. (2021), for example, evaluate 18 different estimators, among which are *Grassberger* (Grassberger, 2008), *Chao Shen* (Chao & Shen, 2003), *NSB* (Nemenman et al., 2001), *Zhang* (Zhang, 2012), and *James-Stein* (Hausser & Strimmer, 2009). These estimators were scattered across 3 different programming languages and 7 different libraries. Some of these estimators are hard to find or poorly maintained. Each implementation calculates and reports entropy to a different number of significant digits, which can lead to difficulties in comparison.

If one can estimate entropy more accurately, then one can also estimate mutual information more accurately. There are numerous, cross-domain, applications for entropy and mutual information, such as in telecommunications, machine learning (MacKay, 2003), and software engineering (Blackwell et al., 2025; Böhme et al., 2020). `DiscreteEntropy.jl` makes it easy to apply different estimators to the problem of mutual information, cross entropy and Kullback–Leibler divergence (Cover & Thomas, 2006), amongst other measures.

`DiscreteEntropy.jl` provides a comprehensive collection of popular entropy estimators and utilities for working with other Shannon measures. `DiscreteEntropy.jl` implements a variety of different entropy estimators, which were previously scattered over different languages and libraries. Some of these scattered implementations are slow, hard to find, or difficult to compile. `DiscreteEntropy.jl` removes all of these problems. `DiscreteEntropy.jl` also provides functions for estimating cross entropy, KL divergence, mutual information and many other Shannon measures. `DiscreteEntropy.jl` is intended to be easy to use, with a flexible but type safe interface.

Statement of need

The `DiscreteEntropy.jl` package has native Julia implementations of all of the estimators explored in Contreras Rodríguez et al. (2021) and a number of estimators which were not considered. `DiscreteEntropy.jl` provides a unified and consistent interface for those who

wish to estimate entropy and other Shannon measures for their research, or those who want to research entropy estimation directly.

There is no other open-source software package known to us, in any language, with similar features or similar breadth of estimators. The R entropy (Hausser & Strimmer, 2009) package covers many basic estimators, such as the maximum likelihood, Miller-Madow, Chao Shen, and many Bayesian estimators. Code for PYM (Pillowlab, 2020), BUB (Paninski), and Unseen (Valiant & Valiant) estimators are found only in the Matlab implementations by the authors of the original papers. Other estimators, such as Zhang and Grassberger, can be found in the R Entropart (Marcon & Hérault, 2015) library. Code for the NSB estimator exists in multiple different versions, in C++ (Nemenman), Matlab (Nemenman), and Python (Marsili, 2021). The estimators in `DiscreteEntropy.jl` were mostly implemented from the original papers, though some (such as BUB and Unseen) are idiomatic ports of original Matlab code.

`DiscreteEntropy.jl` is a fast, simple to use library that fills a gap between the scattered implementations available online. It ensures type safety throughout, even preventing confusion between vectors of samples or vectors which represent histograms of samples.

Example

`DiscreteEntropy.jl` allows the user to call each estimator directly, or to use a helper function `estimate_h`. The `estimate_h` function is the easiest entry to the library. This function takes a `CountData` object, which can be constructed from a vector using either `from_data`, `from_counts` or `from_samples`. Both `from_data` and `estimate_h` are parameterised by types, making them both typesafe and allowing for simple autocompletion. All results are in `nats`, but `DiscreteEntropy.jl` provides helper functions to convert between units.

```
data = [1,2,3,4,5,4,3,2,1]
count_data = from_data(data, Histogram)

estimate_h(count_data, MaximumLikelihood)
2.078803548653078
```

Unsurprisingly, different estimators give different results, depending on their underlying assumptions:

```
estimate_h(count_data, ChaoShen)
2.2526294444274044
```

These assumptions can have a profound effect on estimations of more complex measures, such as mutual information:

```
to_bits(mutual_information(Matrix([1 0; 0 1]), MaximumLikelihood))
1.0

to_bits(mutual_information(Matrix([1 0; 0 1]), ChaoWangJost))
1.7548875021634693
```

Acknowledgements

David Kelly was funded by UKRI Trustworthy Autonomous Systems Hub (reference EP/V00784X/1) and Trustworthy Autonomous Systems Node in Verifiability (reference EP/V026801/2). Ilaria La Torre was partially supported by Meta's Flaky project.

References

Basharin, G. P. (1959). On a statistical estimate for the entropy of a sequence of independent random variables. *Theory of Probability & Its Applications*, *4*(3), 333–336. https://doi.org/10.1137/1104033

Blackwell, D., Becker, I., & Clark, D. (2025). Hyperfuzzing: Black-box security hypertesting with a grey-box fuzzer. *Empirical Software Engineering*, *30*(1), 1–28. https://doi.org/10.1007/s10664-024-10556-3

Böhme, M., Manès, V., & Cha, S. K. (2020). Boosting fuzzer efficiency: An information theoretic perspective. *Proceedings of the 14th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 970–981. https://doi.org/10.1145/3368089.3409748

Chao, A., & Shen, T.-J. (2003). Nonparametric estimation of Shannon's diversity index when there are unseen species in sample. Environ ecol stat 10: 429-443. *Environmental and Ecological Statistics*, *10*, 429–443. https://doi.org/10.1023/A:1026096204727

Contreras Rodríguez, L., Madarro-Capó, E. J., Legón-Pérez, C. M., Rojas, O., & Sosa-Gómez, G. (2021). Selecting an effective entropy estimator for short sequences of bits and bytes with maximum entropy. *Entropy*, *23*(5), 561. https://doi.org/10.3390/e23050561

Cover, T. M., & Thomas, J. A. (2006). *Elements of information theory 2nd edition (Wiley series in telecommunications and signal processing)*. Hardcover; Wiley-Interscience. ISBN: 0471241954

Grassberger, P. (2008). *Entropy estimates from insufficient samplings*. https://arxiv.org/abs/physics/0307138

Hausser, J., & Strimmer, K. (2009). *Entropy inference and the James-Stein estimator, with application to nonlinear gene association networks*. https://arxiv.org/abs/0811.3579

MacKay, D. J. C. (2003). *Information theory, inference, and learning algorithms*. Cambridge University Press. https://doi.org/10.1109/tit.2004.834752

Marcon, E., & Hérault, B. (2015). entropart: An R package to measure and partition diversity. *Journal of Statistical Software*, *67*(8), 1–26. https://doi.org/10.18637/jss.v067.i08

Marsili, S. (2021). ndd - Bayesian entropy estimation from discrete data. In *GitHub repository*. GitHub. https://github.com/simomarsili/ndd

Nemenman, I. *NSB entropy estimation*. https://sourceforge.net/projects/nsb-entropy/ [Accessed 2024-10-24].

Nemenman, I., Shafee, F., & Bialek, W. (2001). Entropy and inference, revisited. *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, 471–478. https://doi.org/10.7551/mitpress/1120.003.0065

Paninski, L. *BUB*. http://www.stat.columbia.edu/~liam/research/code/BUBfunc.m [Accessed 2024-10-22].

Pillowlab. (2020). PYM entropy estimator MATLAB reference implementation. In *GitHub repository*. GitHub. https://github.com/pillowlab/PYMentropy/

Valiant, P., & Valiant, G. *Unseen*. https://theory.stanford.edu/~valiant/code.html [Accessed 2024-10-24].

Zhang, Z. (2012). Entropy estimation in Turing's perspective. *Neural Computation*, *24*(5), 1368–1389. https://doi.org/10.1162/NECO_a_00266