Package: RenyiExtropy (via r-universe)

May 17, 2026

Type Package

Title Entropy and Extropy Measures for Probability Distributions

Version 0.4.0

Description Provides functions to compute Shannon entropy, Renyi entropy, Tsallis entropy, and related extropy measures for discrete probability distributions. Includes joint and conditional entropy, KL divergence, Jensen-Shannon divergence, cross-entropy, normalized entropy, and Renyi extropy (including the conditional and maximum forms). All measures use the natural logarithm (nats). Useful for information theory, statistics, and machine learning applications.

License MIT + file LICENSE

URL https://github.com/itsmdivakaran/RenyiExtropy

BugReports https://github.com/itsmdivakaran/RenyiExtropy/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Repository https://itsmdivakaran.r-universe.dev

Date/Publication 2026-03-17 06:29:09 UTC

RemoteUrl https://github.com/itsmdivakaran/renyiextropy

RemoteRef HEAD

RemoteSha 92bb6603a09a4feb7bde9794d87a4ec520927303

Contents

`conditional_entropy` *Conditional Entropy*

Description

Computes the conditional Shannon entropy $H(Y \mid X)$ for two discrete random variables given their joint probability matrix.

Usage

```
conditional_entropy(joint_matrix)
```

Arguments

`joint_matrix`	A numeric matrix of joint probabilities, where entry (i, j) gives $P(X = i, Y = j)$. All entries must be non-negative and the matrix must sum to 1. Must not contain `NA` or `NaN` values.

Details

The conditional entropy is computed via the chain rule:

$$H(Y \mid X) = H(X, Y) - H(X)$$

where $H(X, Y)$ is the joint entropy and $H(X)$ is the marginal entropy of X (obtained by summing over rows of `joint_matrix`).

The conditional entropy satisfies $0 \leq H(Y \mid X) \leq H(Y)$, with equality on the right when X and Y are independent.

Value

A single non-negative numeric value giving the conditional entropy $H(Y \mid X)$ in nats.

See Also

joint_entropy(), shannon_entropy(), conditional_renyi_extropy()

Examples

```
# 2 x 2 joint distribution
Pxy <- matrix(c(0.2, 0.3, 0.1, 0.4), nrow = 2, byrow = TRUE)
conditional_entropy(Pxy)

# Independent variables: H(Y|X) = H(Y)
px <- c(0.4, 0.6)
py <- c(0.3, 0.7)
Pxy_indep <- outer(px, py)
conditional_entropy(Pxy_indep)
shannon_entropy(py)
```

---

conditional_renyi_extropy *Conditional Renyi Extropy*

Description

Computes the conditional Renyi extropy $J_q(Y \mid X)$ for two discrete random variables given their joint probability matrix.

Usage

```
conditional_renyi_extropy(Pxy, q)
```

Arguments

Pxy	A numeric matrix representing the joint probability distribution, where entry (i, j) gives $P(X = i, Y = j)$. Rows correspond to values of X and columns to values of Y. All entries must be non-negative and the matrix must sum to 1. Must not contain NA or NaN values.
q	A single positive numeric value giving the Renyi order parameter. Must satisfy $q > 0$. When $\lvert q - 1\rvert < 10^{-8}$, the limiting (Shannon) conditional extropy is returned.

Details

The conditional Renyi extropy is defined via the chain rule:

$$J_q(Y \mid X) = J_q(X, Y) - J_q(X)$$

where $J_q(\cdot)$ denotes the Renyi extropy computed on the joint and marginal distributions respectively.

As $q \to 1$, this converges to the conditional Shannon extropy.

Value

A single numeric value giving the conditional Renyi extropy $J_q(Y \mid X)$ in nats.

See Also

renyi_extropy(), conditional_entropy(), max_renyi_extropy()

Examples

```
# 2 x 2 joint distribution
Pxy <- matrix(c(0.2, 0.3, 0.1, 0.4), nrow = 2, byrow = TRUE)
conditional_renyi_extropy(Pxy, q = 2)

# Limiting case q -> 1 (Shannon conditional extropy)
conditional_renyi_extropy(Pxy, q = 1 + 1e-9)

# 3 x 3 joint distribution
Pxy3 <- matrix(c(0.1, 0.05, 0.15, 0.05, 0.2, 0.1, 0.1, 0.15, 0.1),
               nrow = 3, byrow = TRUE)
conditional_renyi_extropy(Pxy3, q = 2)
```

---

cross_entropy *Cross-Entropy*

Description

Computes the cross-entropy between a true probability distribution P and an approximating distribution Q.

Usage

```
cross_entropy(p, q)
```

Arguments

p
: A numeric vector of probabilities representing the true (reference) distribution. Elements must lie in $[0, 1]$, sum to 1, contain at least 2 elements, and must not contain `NA` or `NaN` values.

q
: A numeric vector of probabilities representing the approximating distribution. Must be the same length as `p`, with elements in $[0, 1]$, summing to 1, and must not contain `NA` or `NaN` values.

Details

The cross-entropy is defined as:

$$H(P, Q) = -\sum_{i=1}^{n} p_i \log q_i$$

Terms where $p_i = 0$ are omitted. If $q_i = 0$ but $p_i > 0$, q_i is replaced with 10^{-15} and a warning is emitted.

Cross-entropy relates to KL divergence via: $H(P, Q) = H(P) + D_{\mathrm{KL}}(P \| Q)$.

Value

A single non-negative numeric value giving the cross-entropy $H(P, Q)$ in nats.

See Also

`kl_divergence()`, `js_divergence()`, `shannon_entropy()`

Examples

```
p <- c(0.2, 0.5, 0.3)
q <- c(0.3, 0.4, 0.3)
cross_entropy(p, q)

# When P == Q, cross-entropy equals Shannon entropy
cross_entropy(p, p)
shannon_entropy(p)
```

`extropy` *Classical Extropy*

Description

Computes the classical extropy for a discrete probability distribution.

Usage

```
extropy(p)
```

Arguments

p — A numeric vector of probabilities $(p_1, \ldots, p_n)$, where each element must lie in $[0, 1]$ and the elements must sum to 1. The vector must contain at least 2 elements and must not contain `NA` or `NaN` values.

Details

The classical extropy is defined as:

$$J(\mathbf{p}) = -\sum_{i=1}^{n} (1 - p_i) \log(1 - p_i)$$

Terms where $1 - p_i = 0$ (i.e., $p_i = 1$) are omitted to avoid $\log 0$. Extropy is the dual complement of entropy and measures information in terms of the complementary probabilities.

Value

A single non-negative numeric value giving the classical extropy (in nats).

See Also

`shannon_entropy()`, `renyi_extropy()`, `shannon_extropy()`

Examples

```
# Three-outcome distribution
p <- c(0.2, 0.5, 0.3)
extropy(p)

# Uniform distribution
extropy(rep(0.25, 4))

# Binary distribution
extropy(c(0.7, 0.3))
```

---

joint_entropy *Joint Entropy*

Description

Computes the joint Shannon entropy for a bivariate discrete distribution specified by a joint probability matrix.

Usage

```
joint_entropy(joint_matrix)
```

Arguments

`joint_matrix` A numeric matrix of joint probabilities, where entry (i, j) gives $P(X = i, Y = j)$. All entries must be non-negative and the matrix must sum to 1. Must not contain `NA` or `NaN` values.

Details

The joint entropy is defined as:

$$H(X, Y) = -\sum_{i,j} P(X = i, Y = j) \log P(X = i, Y = j)$$

with the convention that $0 \log 0 = 0$. This satisfies $H(X, Y) \geq \max(H(X), H(Y))$ and the chain rule $H(X, Y) = H(X) + H(Y \mid X)$.

Value

A single non-negative numeric value giving the joint entropy $H(X, Y)$ in nats.

See Also

`conditional_entropy()`, `shannon_entropy()`

Examples

```
# 2 x 2 joint distribution
Pxy <- matrix(c(0.2, 0.3, 0.1, 0.4), nrow = 2, byrow = TRUE)
joint_entropy(Pxy)

# Independent distributions: H(X,Y) = H(X) + H(Y)
px <- c(0.4, 0.6)
py <- c(0.3, 0.7)
Pxy_indep <- outer(px, py)
joint_entropy(Pxy_indep)
shannon_entropy(px) + shannon_entropy(py)
```

`js_divergence` *Jensen-Shannon Divergence*

Description

Computes the Jensen-Shannon (JS) divergence between two probability distributions.

Usage

```
js_divergence(p, q)
```

Arguments

p	A numeric vector of probabilities. Elements must lie in $[0, 1]$, sum to 1, contain at least 2 elements, and must not contain NA or NaN values.
q	A numeric vector of probabilities. Must be the same length as p, with elements in $[0, 1]$, summing to 1, and must not contain NA or NaN values.

Details

The JS divergence is defined as:

$$\mathrm{JSD}(P\|Q) = \frac{1}{2}D_{\mathrm{KL}}(P\|M) + \frac{1}{2}D_{\mathrm{KL}}(Q\|M)$$

where $M = (P + Q)/2$ is the mixture distribution and D_{KL} is the KL divergence.

Unlike kl_divergence(), the JS divergence is symmetric and always finite (even when one distribution has zero entries). Its square root is a metric.

Value

A single non-negative numeric value giving the Jensen-Shannon divergence in nats. The value is bounded in $[0, \log 2]$.

See Also

kl_divergence(), cross_entropy()

Examples

```
p <- c(0.2, 0.5, 0.3)
q <- c(0.3, 0.4, 0.3)
js_divergence(p, q)

# Symmetry: JSD(P, Q) == JSD(Q, P)
js_divergence(q, p)

# JSD is 0 when P == Q
js_divergence(p, p)

# JSD <= log(2) for any two distributions
js_divergence(c(1, 0), c(0, 1))
log(2)
```

kl_divergence *Kullback-Leibler Divergence*

Description

Computes the Kullback-Leibler (KL) divergence from distribution Q to distribution P.

Usage

```
kl_divergence(p, q)
```

Arguments

p	A numeric vector of probabilities representing the true (reference) distribution. Elements must lie in $[0, 1]$, sum to 1, contain at least 2 elements, and must not contain NA or NaN values.
q	A numeric vector of probabilities representing the approximating distribution. Must be the same length as p, with elements in $[0, 1]$, summing to 1, and must not contain NA or NaN values.

Details

The KL divergence is defined as:

$$D_{\mathrm{KL}}(P\|Q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$$

Terms where $p_i = 0$ are omitted (convention $0 \log 0 = 0$). If $q_i = 0$ but $p_i > 0$ the divergence is technically infinite; this function replaces such q_i with 10^{-15} and emits a warning.

Note that KL divergence is not symmetric: $D_{\mathrm{KL}}(P\|Q) \neq D_{\mathrm{KL}}(Q\|P)$ in general. For a symmetric measure see js_divergence().

Value

A single non-negative numeric value giving the KL divergence $D_{\mathrm{KL}}(P\|Q)$ in nats.

See Also

js_divergence(), cross_entropy(), shannon_entropy()

Examples

```
p <- c(0.2, 0.5, 0.3)
q <- c(0.3, 0.4, 0.3)
kl_divergence(p, q)

# KL divergence is zero when P == Q
kl_divergence(p, p)
```

```
# Asymmetry: KL(P||Q) != KL(Q||P)
kl_divergence(p, q)
kl_divergence(q, p)
```

`max_renyi_extropy` *Maximum Renyi Extropy*

Description

Computes the maximum Renyi extropy, which is attained by the uniform discrete distribution over n outcomes.

Usage

```
max_renyi_extropy(n)
```

Arguments

n	A single integer value giving the number of outcomes. Must satisfy $n \geq 2$.

Details

For a uniform distribution $p_i = 1/n$, the maximum Renyi extropy is:

$$\max_{\mathbf{p}} J_q(\mathbf{p}) = (n-1)\log\left(\frac{n}{n-1}\right)$$

Remarkably, this value is independent of the Renyi parameter q (for any $q > 0$, $q \neq 1$).

Value

A single positive numeric value giving the maximum Renyi extropy.

See Also

renyi_extropy(), conditional_renyi_extropy()

Examples

```
# Maximum Renyi extropy for n = 3 outcomes
max_renyi_extropy(3)

# For n = 10 outcomes
max_renyi_extropy(10)

# Verify against renyi_extropy() with uniform distribution
max_renyi_extropy(4)
renyi_extropy(rep(0.25, 4), q = 2)
```

`normalized_entropy` *Normalized Shannon Entropy*

Description

Computes the normalized Shannon entropy (also called relative entropy or efficiency) for a discrete probability distribution.

Usage

```
normalized_entropy(p)
```

Arguments

`p`	A numeric vector of probabilities $(p_1, \ldots, p_n)$, where each element must lie in $[0, 1]$ and the elements must sum to 1. The vector must contain at least 2 elements and must not contain `NA` or `NaN` values.

Details

The normalized entropy is defined as:

$$H_{\mathrm{norm}}(\mathbf{p}) = \frac{H(\mathbf{p})}{\log n}$$

where $H(\mathbf{p})$ is the Shannon entropy and n is the number of outcomes. The result equals 0 for a degenerate distribution and 1 for a uniform distribution.

Value

A single numeric value in $[0, 1]$ giving the ratio of the Shannon entropy to its maximum possible value $\log n$.

See Also

`shannon_entropy()`

Examples

```
# Uniform distribution: normalized entropy = 1
normalized_entropy(rep(0.25, 4))

# Degenerate distribution: normalized entropy = 0
normalized_entropy(c(1, 0, 0))

# Intermediate distribution
normalized_entropy(c(0.2, 0.5, 0.3))
```

`renyi_entropy` *Renyi Entropy*

Description

Computes the Renyi entropy for a discrete probability distribution and order parameter q.

Usage

```
renyi_entropy(p, q)
```

Arguments

- `p` A numeric vector of probabilities $(p_1, \ldots, p_n)$, where each element must lie in $[0, 1]$ and the elements must sum to 1. The vector must contain at least 2 elements and must not contain `NA` or `NaN` values.
- `q` A single positive numeric value giving the Renyi order parameter. Must satisfy $q > 0$. When q is numerically close to 1 (within 10^{-8}), the Shannon entropy is returned instead.

Details

The Renyi entropy of order q is defined as:

$$H_q(\mathbf{p}) = \frac{1}{1-q} \log \left(\sum_{i=1}^{n} p_i^q \right)$$

For $q \to 1$, this reduces to the Shannon entropy:

$$H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log p_i$$

The function detects when $|q - 1| < 10^{-8}$ and returns the Shannon entropy in that case to avoid numerical issues near the singularity.

Special cases: H_0 is the log of the support size (Hartley entropy), H_2 is the collision entropy, and H_∞ is the min-entropy.

Value

A single numeric value giving the Renyi entropy (in nats).

See Also

`shannon_entropy()`, `tsallis_entropy()`, `renyi_extropy()`

Examples

```
p <- c(0.2, 0.5, 0.3)

# Renyi entropy for q = 2 (collision entropy)
renyi_entropy(p, 2)

# Renyi entropy for q = 0.5
renyi_entropy(p, 0.5)

# q near 1 returns Shannon entropy
renyi_entropy(p, 1 + 1e-9)

# Explicit Shannon entropy for comparison
shannon_entropy(p)
```

`renyi_extropy` *Renyi Extropy*

Description

Computes the Renyi extropy for a discrete probability distribution and Renyi order parameter q.

Usage

```
renyi_extropy(p, q)
```

Arguments

p	A numeric vector of probabilities $(p_1, \ldots, p_n)$, where each element must lie in $[0, 1]$ and the elements must sum to 1. The vector must contain at least 2 elements and must not contain `NA` or `NaN` values.
q	A single positive numeric value giving the Renyi order parameter. Must satisfy $q > 0$. When $|q-1| < 10^{-8}$, the classical extropy (Shannon extropy) is returned instead.

Details

The Renyi extropy of order q is defined as:

$$J_q(\mathbf{p}) = \frac{-(n-1)\log(n-1) + (n-1)\log(\sum_{i=1}^{n}(1-p_i)^q)}{1-q}$$

where n is the number of outcomes.

For $q \to 1$, the formula reduces to the classical (Shannon) extropy:

$$J(\mathbf{p}) = -\sum_{i=1}^{n}(1-p_i)\log(1-p_i)$$

For $n = 2$, the Renyi extropy coincides with the Renyi entropy.

Value

A single numeric value giving the Renyi extropy (in nats).

See Also

extropy(), renyi_entropy(), conditional_renyi_extropy(), max_renyi_extropy()

Examples

```
p <- c(0.2, 0.5, 0.3)

# Renyi extropy for q = 2
renyi_extropy(p, 2)

# For q near 1, returns classical extropy
renyi_extropy(p, 1 + 1e-9)

# Compare with extropy() at the limiting case
extropy(p)

# Binary distribution (n = 2): Renyi extropy equals Renyi entropy
renyi_extropy(c(0.4, 0.6), 2)
renyi_entropy(c(0.4, 0.6), 2)
```

---

shannon_entropy *Shannon Entropy*

Description

Computes the Shannon entropy for a discrete probability distribution.

Usage

```
shannon_entropy(p)
```

Arguments

p	A numeric vector of probabilities $(p_1, \ldots, p_n)$, where each element must lie in $[0, 1]$ and the elements must sum to 1. The vector must contain at least 2 elements and must not contain `NA` or `NaN` values.

Details

The Shannon entropy is defined as:

$$H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log(p_i)$$

with the convention that $0 \log 0 = 0$ (terms where $p_i = 0$ contribute zero). The logarithm used is the natural logarithm, so the result is in nats.

Shannon entropy is the limiting case of the Renyi entropy as $q \to 1$ and equals zero if and only if the distribution is degenerate. It is maximised by the uniform distribution, where $H = \log n$.

Value

A single non-negative numeric value giving the Shannon entropy measured in nats (natural-logarithm base).

See Also

`renyi_entropy()`, `tsallis_entropy()`, `extropy()`, `normalized_entropy()`

Examples

```
# Three-outcome distribution
p <- c(0.2, 0.5, 0.3)
shannon_entropy(p)

# Uniform distribution -- maximum entropy equals log(n)
shannon_entropy(rep(0.25, 4))

# Degenerate distribution -- entropy equals 0
shannon_entropy(c(1, 0, 0))
```

`shannon_extropy` *Shannon Extropy*

Description

Computes the Shannon extropy for a discrete probability distribution.

Usage

```
shannon_extropy(p)
```

Arguments

`p`	A numeric vector of probabilities $(p_1, \ldots, p_n)$, where each element must lie in $[0, 1]$ and the elements must sum to 1. The vector must contain at least 2 elements and must not contain `NA` or `NaN` values.

Details

Shannon extropy is defined as:

$$J(\mathbf{p}) = -\sum_{i=1}^{n} (1 - p_i) \log(1 - p_i)$$

Terms where $p_i = 1$ are treated as contributing zero (using the convention $0 \log 0 = 0$).

This function is an alias for extropy(); both compute the same quantity. It is retained for naming consistency with shannon_entropy().

Value

A single non-negative numeric value giving the Shannon extropy (in nats).

See Also

extropy(), shannon_entropy(), renyi_extropy()

Examples

```
p <- c(0.2, 0.5, 0.3)
shannon_extropy(p)

# Identical to extropy()
extropy(p)

# Uniform distribution
shannon_extropy(rep(1/3, 3))
```

---

tsallis_entropy *Tsallis Entropy*

Description

Computes the Tsallis entropy for a discrete probability distribution and order parameter q.

Usage

```
tsallis_entropy(p, q)
```

Arguments

p	A numeric vector of probabilities $(p_1, \ldots, p_n)$, where each element must lie in $[0, 1]$ and the elements must sum to 1. The vector must contain at least 2 elements and must not contain NA or NaN values.
q	A single positive numeric value giving the Tsallis order parameter. Must satisfy $q > 0$. When $\lvert q - 1 \rvert < 10^{-8}$, the Shannon entropy is returned.

Details

The Tsallis entropy of order q is defined as:

$$S_q(\mathbf{p}) = \frac{1 - \sum_{i=1}^{n} p_i^q}{q - 1}$$

For $q \to 1$, this reduces to the Shannon entropy:

$$H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log p_i$$

The Tsallis entropy is non-extensive and is widely used in non-equilibrium statistical mechanics and complex systems.

Value

A single non-negative numeric value giving the Tsallis entropy.

See Also

`shannon_entropy()`, `renyi_entropy()`

Examples

```
p <- c(0.2, 0.5, 0.3)

# Tsallis entropy for q = 2
tsallis_entropy(p, 2)

# Tsallis entropy for q = 0.5
tsallis_entropy(p, 0.5)

# As q -> 1, converges to Shannon entropy
tsallis_entropy(p, 1 + 1e-9)
shannon_entropy(p)
```

Index