Package 'queeems'

November 12, 2026

Version 1.0.0

Date 2025-04-28

License GPL-3 + file LICENSE

Title Quantify the Extent of Evolutionary Evidence in Molecular Sequences

Description Biological inferences obtained from molecular data are only as good as the extent of evolutionary signatures retained in the genetic data. Techniques available to quantify these signatures are largely targeted towards phylogeny reconstruction and they often rely on adhoc hypothesis tests of significance. I present a Bayesian function that assesses whether a set of genetic sequences are saturated. That is, it is useful for determining whether the evolutionary information in the sequences has eroded with time. Site specific Bayes factors are generated with respect to codon bases to allow for straightforward applications in extensive computational biology inquiries, including natural selection analyses.

Depends R (>= 4.6.0), Biostrings

Imports gtools, Matrix, methods, stats

Suggests BiocStyle, knitr, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

biocViews Alignment, Bayesian, Classification, DataImport, Genetics, MathematicalBiology, ResearchField, Sequencing, SequenceMatching, Software, StatisticalMethod, WorkflowStep

Contact <hassan.t.sadiq@gmail.com>

URL https://github.com/thsadiq/queeems

BugReports https://github.com/thsadiq/queeems/issues

git_url https://git.bioconductor.org/packages/queeems

git_branch RELEASE_3_23

git_last_commit c33f208

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Hassan Sadiq [aut, cre, cph, fnd] (ORCID:
<https://orcid.org/0000-0003-0192-7134>)

Maintainer Hassan Sadiq <hassan.t.sadiq@gmail.com>

Contents

aboutQueeems . . . 2
baseFrequency . . . 3
baseSummary-class . . . 4
bstringCodons . . . 5
citing-class . . . 6
CnCs . . . 7
cncsentropy . . . 8
cncsframe-class . . . 10
codondifferindex . . . 11
codonDissimilarity . . . 12
diffNucBinary . . . 13
entropyindex . . . 14
fubarweights . . . 15
geneindex-class . . . 16
mnomLogl . . . 18
molentropy . . . 19
n2cFreqs . . . 20
nonSynonymous . . . 21
queeems . . . 22
queeemsExtdata . . . 23
saturateBF-class . . . 24
seqfilter . . . 26
seqSaturation . . . 27
siteindices-class . . . 30
softmax . . . 32

Index **34**

---

aboutQueeems *References Relevant to the* queeems *Package*

Description

Obtain citations of research outputs that may assist with understanding and extending the applications of the queeems package.

Usage

```
aboutQueeems(cite=NULL)
```

Arguments

cite	A strictly positive integer.

Details

Citations included are with respect to outputs that significantly use the queeems package and that the package author contributes to. The list will be extended as more outputs are published. When cite = NULL is set (default), all the available citations are listed. If the integer input for cite exceeds the number of available citations, an error message is returned.

Value

A text that contains citations associated with the queeems package, as explained in **Details**.

Author(s)

Hassan Sadiq

Examples

```
aboutQueeems()
```

---

baseFrequency *Counts Indicating Protein Base Prevalence*

Description

Obtain, from empirical molecular data, frequencies of each base (DNA, codon or amino acid) per data site.

Usage

```
baseFrequency(fastafile, basename="dna")
```

Arguments

fastafile	path to a FASTA format file that contains a protein sequence.
basename	identification of the protein constituent base.

Details

basename should be a character input and one of ‘dna‘ (defualt), ‘codon‘ or ‘aa‘. Any other input will trigger an error. A warning message may also be returned if there is an indication of sequence-basename mismatch. For example if ‘dna‘ is specified for a protein sequence saved in terms of amino acid bases.

Value

A baseSummary object that contains a matrix of site-wise base frequencies that are labelled with the corresponding base notation.

Author(s)

Hassan Sadiq

See Also

The baseSummary class for further description of the output and other similar fuctions such as CnCs and diffNucBinary. Sample FASTA files may also be accessed using the queeemsExtdata function.

Examples

```
testData <- queeemsExtdata("PI.fasta")
tested <- baseFrequency(testData, "codon")
tested
```

---

baseSummary-class *Protein Base Composition Numerics*

Description

Ensure that numerical extracts from protein data are stored and handled elegantly.

Objects from the Class

Objects of this (baseSummary) class can be created by calls of the form new("baseSummary", baseType, extantSize, seqsNumerics).

Slots

baseType: description of the bases that make up the corrsponding protein sequence.

extantSize: number of sequences in the protein data.

seqsNumerics: numerical data that describe the distribution of the primary bases in the captured protein sequences.

Methods

show: signature(object="baseSummary") useful output highlights.

nseqs: signature(bsumm="baseSummary") number of protein sequences.

datatype: signature(bsumm="baseSummary") how protein data positions must be interpreted.

basecensus: signature(bsumm="baseSummary") numerical summary of the empirical protein base distribution.

Details

The baseType slot should be one of `aa`, `codon` or `dna` depending on whether the protein sequence sites must be analysed as 'amino acid', 'codon' or 'DNA' positions. The rows of the element of the seqsNumerics slot are defined with respect to the alphabetically ordered IUPAC abbreviations of the corresponding baseType input. Each column of the matrix represents the relevant protein site.

Author(s)

Hassan Sadiq

See Also

The baseFrequency function that returns baseSummary objects.

Examples

```
proteinFile <- queeemsExtdata("PI.fasta")
testEG <- baseFrequency(proteinFile, "codon")
head( basecensus(testEG))[,seq(1,6)]
datatype(testEG)
nseqs(testEG)
testEG
```

---

bstringCodons *Convert a BStringSet Object to Codon Sequences*

Description

Converts a Biostrings::BStringSet object to a matrix of codons.

Usage

```
bstringCodons(bstringset)
```

Arguments

bstringset	A BStringSet object.

Value

A matrix of codon bases. Each column of the matrix will contain the site data and each row will hold the respective sequence that make up the input protein data.

Author(s)

Hassan Sadiq

Examples

```
dataPath <- queeemsExtdata("PI.fasta")
bstringdata <- readBStringSet(dataPath)
testing <- bstringCodons(bstringdata)
head(testing)[,seq(1,6)]
```

`citing-class` *Useful Package Citations*

Description

Get details of citations that may further assist with understanding of package utility.

Objects from the Class

Objects of this (`citing`) class can be created by calls of the form `new("citing", citeText=...)`.

Slots

`citeText`**:** printable text that may be pasted directly as references into scientific write-ups.

Methods

show: `signature(object = "citing")` prints the input text.

Author(s)

Hassan Sadiq

See Also

The `aboutQueeems` function that relies on the described class.

Examples

```
test <- aboutQueeems()
is(test, "citing")
test
```

CnCs *Synonymous and Non-Synonymous Codon Match Counts*

Description

Crude counts of the number of synonymous or non-synonymous matches at each codon site of input empirical molecular data.

Usage

```
CnCs(bstringset, synonym=FALSE, diffceil=1)
```

Arguments

bstringset	Biostrings::Bstringset object that carries an empirical protein sequence.
synonym	logical entry that indicates whether synonymous [TRUE] or non-synonymous [FALSE (default)] counts are needed.
diffceil	maximum number of nucleotide mismatches that is tolerated.

Details

Each element of the nscensus output provides information about the respective codon site. Consider the case where synonym = TRUE. At each site, the most frequent codon is identified as the wildtype base. The number of spots at the site that are filled by separate but synonymous codons to the wild-type is recorded. Ambiguities and/or missing observations are ignored. Sites with zero counts are considered non-informative. Observe that such sites are not necessarily invariant. For example, a site with three unique non-synonymous codon bases will only be informative when non-synonymous counts are sought, i.e. if synonym = FALSE. The diffceil input implies the maximum, not the exact, number of nucleotide mismatches allowed. For example, diffceil = 1 (default) accommodates zero or one nucleotide mismatches.

Value

A cncsframe object that contains the following.

nscensus	an array of integers that may be accessed with the nsfreqs function. They represent site-wise counts, compared to the identified wild-type, of [non-]synonymous occurrences.
wildseq	an array of the most frequent, i.e. the identified wild-type, codon at each site of the input molecular data.
nstype	a string that describes whether the entries of nscensus were obtained with respect to synonymous or non-synonymous counts.
maxdiff	maximum number of nucleotide mismatches tolerated when comparing to the wild-type.
invariants	indices of invariant codon sites, after ambiguities and missingness have been accommodated.

Author(s)

Hassan Sadiq

See Also

The cncsframe class that contains additional informationn about how to access the output object. The nonSynonymous and codonDissimilarity functions may be useful resources for clarifications about the maxdiff entry.

Examples

```
indata <- c("CCTCAGATCACTCTTTGGCAACGACCCCTT",
            "CCTCAGATCACTCTTGTCTCAATAAAAGTA",
            "TGG-AGCGACCCCTTGTCTCAATAAAAATA")
codonarray <- Biostrings::BStringSet(indata)
tested <- CnCs(codonarray, FALSE, 3)
tested
```

---

cncsentropy *Global [Non-]Synonymous Entropy Index for Protein Sequence*

Description

Obtain homogeneous entropy estimate for a set of homologous protein sequences, based on observed synonymous or non-synonymous codon diversities.

Usage

```
cncsentropy(fastafile, synonym=FALSE, diffceil=1, fml="shannon", ralpha=NA)
```

Arguments

fastafile	path to a FASTA format file that contains a protein sequence.
synonym	logical entry that indicates whether synonymous [TRUE] or non-synonymous [FALSE (default)] counts are preferred.
diffceil	maximum number of nucleotide mismatches that is tolerated. Default value is 1.
fml	character input that can be one of "shannon" (default) or "renyi".
ralpha	a fraction between 0 and 1 that is needed for Renyi index calculation.

Details

Entropy calculations are largely functions of relative counts. See the entropyindex function for the mathematical expressions and some useful citations. Obtaining site-specific entropy values, where the proportions represent the relative occurrence of the protein bases per site, is not ambiguous. With respect to obtaining a single entropy estimate for an entire alignment in terms of synonymous and non-synonymous relationships, the process is less straightforward. The proportions used for the calculations were obtained as follows.

$$p_i^{\text{syn}} = \frac{n_i^{\text{syn}}}{\sum_{i=1}^{k} n_i^{\text{syn}}},$$

where p_i^{syn} represents the proportion of observed synonymous substitutions at site $i = 1, 2, \ldots, k$ and n_i^{syn} is the number of observed synonymous substitutions with reference to the most frequently occurring codon at the site. For example, the expression for the corresponding synonymous Shannon entropy for a protein sequence alignment then becomes,

$$H^{\text{syn}} = -\sum_{i=1}^{k} p_i^{\text{syn}} \times \log p_i^{\text{syn}}.$$

Similar explanation holds for estimates obtained with respect to non-synonymous substitutions or from using the Renyi approach.

Value

A geneindex object.

Author(s)

Hassan Sadiq

References

Shannon, C. E. (1948). A Mathematical Theory of Communication, *The Bell System Technical Journal* **27**(3): 379-423.

See Also

For further clarification about the operation of the cncsentropy function and extensive understanding of the interpretation of its output, consult the constituent CnCs and entropyindex functions. There are also the codondifferindex and molentropy functions that are capable of generating site-wise indices without distinguishing between synonymous or non-synonymous codon relationships.

Examples

```
sqdatm <- c("CCTCAGATCACTCTTTGGCAACGACCCCTT",
            "CCTCAGATCACTCTTGTCTCAATAAAAGTA",
            "CCC-AGCGACCCCTTGTCTCAATAAAAATA")
codonData <- Biostrings::BStringSet(sqdatm)
names(codonData) <- c("s1","s2","s3")
```

```
testdna <- file.path(tempdir(), "testDna.fasta")
Biostrings::writeXStringSet(codonData, testdna)
testing <- cncsentropy(testdna, TRUE, 2, "shannon")
print(testing)
```

`cncsframe-class` *Synonymous and Non-Synonymous Frequencies*

Description

Facilitate smooth extraction and utility of outputs from synonymous and non-synonymous count analyses of molecular protein data.

Objects from the Class

Objects of this (`cncsframe`) class can be created by calls of the form `new("cncsframe", nscensus, wildseq, nstype, maxdiff, invariants)`.

Slots

`nscensus`**:** site-wise counts of codons that are synonymous or non-synonymous to the wild-type codon.

`wildseq`**:** sequence that comprise of the most frequent codon at each site.

`nstype`**:** indication of the type, i.e. `synonymous` or `non-synonymous`, of counts returned.

`maxdiff`**:** maximum number of nucleotide mismatch that is tolerated.

`invariants`**:** indices of the non-informative codon sites.

Methods

show: `signature(object="cncsframe")` snapshot of output features.

nORs: `signature(cNS="cncsframe")` computational approach, ‘`synonymous`‘ or ‘`non-synonymous`‘.

nsfreqs: `signature(cNS="cncsframe")` site-wise counts of codons that are (non-)synonymous to the wild-type.

nucbalance: `signature(cNS="cncsframe")` maximum number of nucleotide mismatch tolerated per codon pair.

wildtriplets: `signature(cNS="cncsframe")` a vector of the identified wild-type codon for each site of the input protein data.

nonvaries: `signature(cNS="cncsframe")` locations of sites that were detected to be non-informative.

Details

Consult the `CnCs` function for more elaborate descriptions of the slots and method outputs.

Author(s)

Hassan Sadiq

See Also

The CnCs function that returns cncsframe objects.

Examples

```
testsq <- c("CCTCAGATCACTCTTTGGCAACGACCCCTT",
            "CCTCAGATCACTCTTGTCTCAATAAAAGTA",
            "TGG-AGCGACCCCTTGTCTCAATAAAAATA")
codonarray <- Biostrings::BStringSet(testsq)
tested <- CnCs(codonarray, FALSE, 3)
wildtriplets(tested)
nsfreqs(tested)
tested
```

---

codondifferindex *Molecular Entropy Estimated from Codon Mismatches*

Description

Obtain information entropy index for each codon site in a protein sequence, as a function of mismatches observed at nucleotide positions.

Usage

```
codondifferindex(fastafile, diffUnit=1, fml="shannon", ralpha=NA)
```

Arguments

fastafile	path to a FASTA format file that contains a protein sequence.
diffUnit	number of nucleotide mismatches between codon pairs to use for the calculations.
fml	character input that can be one of "shannon" (default) or "renyi".
ralpha	a fraction between 0 and 1 that is needed for Renyi index calculation.

Value

An object of class siteindices.

Author(s)

Hassan Sadiq

See Also

This function is a combination of the operations of the codonDissimilarity and the entropyindex functions. The two referenced functions contain further discussions about the diffUnit, fml and ralpha inputs. See the siteindices class for an extensive detail about how to work with the output object. The molentropy function is an example of an alternative approach of estimating site-wise entropy indices accessible within the queeems package.

Examples

```
proteinFile <- queeemsExtdata("VertCOI.fasta")
testEG <- codondifferindex(proteinFile, 1, "shannon")
summary(testEG)
print(testEG)
```

---

codonDissimilarity *Codon Census Matrix Matched by Nucleotide Dissimilarity*

Description

Generates a matrix that contains numerical details of the distribution of bases at each codon position in a protein sequence.

Usage

```
codonDissimilarity(bstringset, diffUnit)
```

Arguments

bstringset	A BStringSet object.
diffUnit	An integer between the 0 and 3 interval.

Details

The input BStringSet object can be created with the Biostrings::readBStringSet function. diffUnit represents the number of nucleotide mismatches that is preferred. Let the unique codons at each genetic sequence alignment site be denoted as i = 1, 2, ..., k, where k can not be more than the number of sequences that make up the data. For each i at the different sites, a value of 1 is recored if any of the other codons at the site differ from codon i by exactly diffUnit nucleotide positions. The process is repeated independently for each site and the outcome is entered into the corresponding column of the output matrix. Note the special case of diffUnit=0. The output will be a binary matrix. Row r column c of the matrix will contain a 0 if codon r was observed at site c of the input protein sequence.

Value

A 61-by-nsites dissimilarity matrix that captures the site-wise distribution of codon bases evident from the input protein data, where nsites is the number of codon sites. As explained as part of **Details** above, codon dissimilarity is determined by specified degree of nucleotide mismatch.

Author(s)

Hassan Sadiq

See Also

queeemsExtdata that returns paths to the example external DNA data that can be processed with the described function. Biostrings::readBStringSet, a necessary feeder function may also be a useful resource. The baseFrequency function implicitly uses this function.

Examples

```
dataPath <- queeemsExtdata("PI.fasta")
bstringdata <- readBStringSet(dataPath)
testing <- codonDissimilarity(bstringdata, 1)
head(testing)[,seq(1,6)]
```

---

diffNucBinary *Compare Codon Pairs to Identify Nucleotide Mismatch*

Description

Generate a binary matrix that identifies pairs of sense codons that differ by a specified degree.

Usage

```
diffNucBinary(diffUnit=1)
```

Arguments

diffUnit	A non-negative integer. Defaults to 1.

Details

diffUnit should ideally be a non-negative value that is less than 4. The codons are matched by nucleotide bases and their positions. For example, comapring codon `ACA` and codon `CAA` will return TRUE only when diffUnit = 2 in accordance with the discrepancies at the first and the second nuleotide positions. Observe that when diffUnit = 0, a logical diagonal matrix is returned.

Value

A 61-by-61 logical matrix. TRUE will appear at the row and column intersection of the sense codons that differ by the number of nucleotide bases specified by diffUnit. FALSE will appear otherwise.

Author(s)

Hassan Sadiq

Examples

```
test1 <- diffNucBinary(1)
head(test1)[,seq(1,6)]
```

`entropyindex` *Obtain Entropy Indices*

Description

Obtain indices that are useful for quantifying the degree of diversity contained in observed molecular prevalence count data.

Usage

```
entropyindex(countvector, nleaf, fml="shannon", ralpha=0.1)
```

Arguments

`countvector`	integer vector of base frequencies.
`nleaf`	number of the extant branches of the tree that generated the data reduced to `countvector`.
`fml`	character input that can be one of `"shannon"` or `"renyi"`.
`ralpha`	a fraction between 0 and 1 that is needed for Renyi index calculation.

Details

Molecular entropy estimation requires knowledge of the size of the leaves on the phylogeny that generated the data. This is expected to be entered into the function as `nleaf`. Let $\boldsymbol{x}$ denote the vector of base frequencies `countvector` and let `n` = `nleaf`. If `fml` is set as `"shannon"`, the diversity index is computed as:

$$H = -\sum_{i=1}^{m} p_i \times \log p_i,$$

where `m` $\leq$ `n` is the length of $\boldsymbol{x}$ after all zeros have been excluded and $\mathtt{p_i} = \mathtt{x_i}/\mathtt{n}$. When `fml` is set as `"renyi"`, then

$$H = \frac{1}{1-\alpha} \log \sum_{i=1}^{m} p_i^{\alpha},$$

where α = `ralpha`.

Value

A numeric estimate of the entropy index obtained for the input data.

Author(s)

Hassan Sadiq

References

Renyi, A. (1961). On Measures of Information and Entropy, *Proceedings of the Fourth Berkeley Symposium on Mathematics, Statistics and Probability 1960* 547-561.

Shannon, C. E. (1948). A Mathematical Theory of Communication, *The Bell System Technical Journal* **27**(3): 379-423.

See Also

Scaled multinomial likelihood estimator `mnomLogl` which returns a scaled entropy. The `seqSaturation` relies heavily on this function, while `baseFrequency`, `CnCs` and `codonDissimilarity` are useful for transforming genetic sequences into counts needed for the `countvector` input. Interested users might consult the external `vegan::diversity` function for alternative entropy estimation methods.

Examples

```
freqdata <- sample(seq(0,6), 20, replace=TRUE)
tested <- entropyindex(freqdata, sum(freqdata))
tested
```

`fubarweights` *FUBAR-like Weights*

Description

Generate unevenly discretised weights

Usage

```
fubarweights(nweights, wless1, weightmax, addzero)
```

Arguments

`nweights`	number of weights.
`wless1`	proportion of weights between the (0, 1) interval.
`weightmax`	maximum weight value.
`addzero`	logical input. `TRUE` if minimum weight value should be zero and `FALSE` otherwise.

Details

For $n > 3$ unevenly discretised weights that are all less than w_{max} such that ϕ of them lie within the (0, 1) interval, the vector of weights is obtained as the following set of values.

$$W = \{w_L, w_U\},$$

such that,

$$w_L = \{\frac{I^{(0)}}{\kappa}, \frac{I^{(0)}+1}{\kappa}, \ldots, \frac{\kappa}{\kappa}\},$$

$$w_U = \{1 + (\delta)^3, 1 + (2\delta)^3, \ldots, 1 + ((n - \kappa + I^{(0)} - 1)\delta)^3\},$$

where

$$\kappa = \lfloor \phi n \rfloor,$$

$$\delta = \frac{\sqrt[3]{w_{max} - 1}}{n - \kappa + I^{(0)} - 1},$$

Further, $I^{(0)}$ equals zero if `addzero = TRUE` and it equals one otherwise, n is the input `weightmax` and ϕ is the specified `wless1`.

Value

A numeric vector of weights.

Author(s)

Hassan Sadiq

References

Murrell, B., Moola, S., Mabona, A., Weighill, T., Sheward, D., Kosakovsky Pond, S. L. and Scheffler, K. (2013) FUBAR: A Fast, Unconstrained Bayesian AppRoximation for Inferring Selection, *Mol. Biol. Evol.* **30**(5): 1196-1205.

Examples

```
wVector1 <- fubarweights(20, 0.70, 50, FALSE)
wVector2 <- fubarweights(20, 0.70, 50, TRUE)
print(wVector1)
print(wVector2)
```

`geneindex-class` *Protein Information Entropy Handling*

Description

Facilitates succinct handling of the global information entropy index estimated for an alignment of homologous protein sequences.

Objects from the Class

A `geneindex` object may be created with a call of the form `new("geneindex", eindex, noinfo, meth, alphaR, nbases, synonym)`.

Slots

`eindex`: single numeric entropy estimates obtained from genetic data.

`noinfo`: numeric vector that contains non-informative positions of the molecular data that produced `eindex`.

`meth`: character entry that informs about the method used to generate `eindex` from the original protein data.

`alphaR`: parameter for Renyi entropy calculations. Only relevant if `meth = "renyi" [default = NA]`.

`nbases`: number of base positions that make up the protein data used to estimate `eindex`.

`synonym`: `default = NA` and only relevant if the counts used to estimate `eindex` are with respect to [non-]synonymous counts.

Methods

show: `signature(object="geneindex")` output highlights.

renyiA: `signature(sent="geneindex")` value of the alpha parameter used for Renyi entropy calculations.

useSyn: `signature(gent="geneindex")` answers the question: were synonymous codon counts used?.

gentropy: `signature(gent="geneindex")` value of the entropy estimate obtained for the genetic sequence.

nonvaries: `signature(cNS="geneindex")` positions of the processed molecular data that are not informative.

informula: `signature(sent="geneindex")` should return the name of the method used to generate the entropy estimate from the original molecular data.

sitecount: `signature(gent="geneindex")` number of base positions that constitute the analysed protein data.

Details

The entry of the `synonym` slot is expected to be logical, if specified. `TRUE` is set if the codon counts used to estimate the `eindex` entry is with respect to synonymous counts and should be `FALSE` if non-synonymous counts are used.

Author(s)

Hassan Sadiq

See Also

- Variant of the `nonvaries` method is applied within the `baseSummary` and the `siteindices` classes.
- The `entropyindex` function is a good reference for discussions about the different entropy calculation techniques accessible from the `queeems` package.
- Functions that return outputs of the described `geneindex-class` include `cncsentropy`.

Examples

```
geneFile <- queeemsExtdata("PI.fasta")
testing <- cncsentropy(geneFile, FALSE, 1, "renyi", 0.04)
print( nonvaries(testing))
print( class(testing))
print(testing)
```

`mnomLogl` *Approximate Log-Likelihood of a Multinomial Variable*

Description

Obtain an approximate estimate of the log-likelihood for a random variable that is believed to have a multinomial distribution.

Usage

```
mnomLogl(countvector, pivector)
```

Arguments

`countvector`	vector of non-negative observed frequencies.
`pivector`	theoretical population proportions that generated the input `countvector`.

Details

Consider a discrere random variable

$$\mathbf{X} = (X_1, \ldots, X_k) \sim \texttt{Multinomial}\,(n,\, \pi_1, \ldots, \pi_k)\,,$$

where, $n = X_1 + \ldots + X_k$ and $\sum_{i=1}^{k} \pi_i = 1$. Then,

$$P\left(\mathbf{x} = (x_1, \ldots, x_k) \mid \boldsymbol{\pi}\right) = \frac{n!}{x_1!, \ldots, x_k!}\, \pi_1^{x_1} \ldots, \pi_k^{x_k}$$

such that the log-likelihood expression may be simplified as follows.

$$l\left(\boldsymbol{\pi} \mid \mathbf{x}\right) = \log\left(n!\right) - \sum_{i=1}^{k} \log\left(x_i!\right) + \sum_{i=1}^{k} x_i \log\left(\pi_i\right) \;\propto\; \sum_{i=1}^{k} x_i \log\left(\pi_i\right).$$

The expression can be shown to produce a maximum likelihod estimate of $\hat{\pi}_i = x_i/n$ by using a Lagrange multiplier that imposes the constraint that $\sum_{i=1}^{k} \pi_i = 1$..

Value

The scaled likelihood value estimated from the final expression presented in **Details**. The output should be a negative value.

Author(s)

Hassan Sadiq

See Also

Some other information retrieval functions within the queeems package that are straightforwardly applicable to molecular protein data. These include, cncsentropy and seqSaturation. Functions such as baseFrequency, CnCs and codonDissimilarity, that are needed to obtain the counts required to successfully execute the described mnomLogl, from a genetic sequences may also be consulted.

Examples

```
arbitraryCounts <- sample(40, 20)
tempPis <- runif(20)
arbitraryPis <- tempPis / sum(tempPis)
tests <- mnomLogl(arbitraryCounts, arbitraryPis)
print(tests)
```

---

molentropy *Molecular Entropy Generated Independently per Base Site*

Description

Generate information entropy estimate independently for protein sequence sites.

Usage

```
molentropy(fastafile, basename="dna", fml="shannon", ralpha=NA)
```

Arguments

fastafile	path to a FASTA format file that contains a protein sequence.
basename	specification of how the data must be encoded. It can be 'dna' (default), 'codon' or 'aa' for DNA, codon or amino acid, respectively.
fml	character input that can be one of "shannon" (default) or "renyi".
ralpha	a fraction between 0 and 1 that is needed for Renyi index calculation.

Value

A siteindices object.

Author(s)

Hassan Sadiq

See Also

For further clarification about the operation of the molentropy function and extensive understanding of the interpretation of its output, consult the baseFrequency and entropyindex functions. There is also the codondifferindex function that can be used to generate site-wise entropy indices and the cncsentropy function that is suitable for homogeneous entropy estimation.

Examples

```
sqdatm <- c("CCTCAGATCACTCTTTGGCAACGACCCCTT",
            "CCTCAGATCACTCTTGTCTCAATAAAAGTA",
            "CCC-AGCGACCCCTTGTCTCAATAAAAATA")
moleculeDta <- Biostrings::BStringSet(sqdatm)
names(moleculeDta) <- c("s1","s2","s3")
testprotein <- file.path(tempdir(), "testprotein.fasta")
Biostrings::writeXStringSet(moleculeDta, testprotein)
test1 <- molentropy(testprotein, "dna", "renyi", .35)
test2 <- molentropy(testprotein, "codon", "shannon")
print(test1)
print(test2)
```

---

n2cFreqs *Generate Codon Frequencies from Nucleotide Frequencies*

Description

Obtain frequencies of sense codons as a function of the frequencies of nucleotide bases

Usage

```
n2cFreqs(nucFracs)
```

Arguments

nucFracs	vector of nucleotide frequencies.

Details

This provides a straightforward way to generate codon frequencies as described for the popular F3x4 model. For a given vector (nucFracs = $\boldsymbol{\pi}^{(n)} = \{\pi_{\mathsf{A}}^{(n)}, \pi_{\mathsf{C}}^{(n)}, \pi_{\mathsf{G}}^{(n)}, \pi_{\mathsf{T}}^{(n)}\}$) of nucleotide frequencies, each non-standardised sense codon frequency, $\pi_i^{(c)}$ for i = 1, 2, ..., 61 is obtained as the product the product of the nucleotide triplet that form the codon. For example, for $\pi_i^{(c)} = \pi_{\mathsf{AGT}}^{(c)}$, the unstandardised codon frequency is estimated as $\pi_{\mathsf{A}}^{(n)} \times \pi_{\mathsf{G}}^{(n)} \times \pi_{\mathsf{T}}^{(n)}$. Please note that, if the contents of the input vector are not named, the elements are assigned base names (A, C, G, T) in order of appearance in the input.

Value

A numeric vector of codon frequencies that are standardised to sum to one, ordered and named in terms of the IUPAC nucleotide triplet nomenclature.

Author(s)

Hassan Sadiq

References

Goldman, N. and Yang, Z. (1994) A Codon-Based Model of Nucleotide Substitution for Protein-Coding DNA Sequences, *Mol. Biol. Evol.* **11**(5): 725-736.

Examples

```
codonFrac1 <- n2cFreqs( c(A=0.3, C=0.1, G=0.4, T=0.2) )
codonFrac2 <- n2cFreqs( c(0.3, 0.1, 0.4, 0.2) )
print(codonFrac1)
print(codonFrac2)
```

---

nonSynonymous *Identify Synonymous and Non-Synonymous Codons*

Description

Create a 61-by-61 symmetric logical matrix that indicates the sense codons that are synonymous or non-synonymous to the corresponding row or column codon.

Usage

```
nonSynonymous(synonym=FALSE, diffceil=1)
```

Arguments

`synonym`	logical input that indicates whether synonymous [TRUE] or non-synonymous [FALSE (default)] identifier is required.
`diffceil`	an interger that states the preferred maximum number of nucleotide mismatch. Set to 1 by default.

Details

`diffceil` should ideally be a non-negative value that is less than 4. The codons are matched by nucleotide bases and their positions. For example, when `diffceil = 2` and `synonym = FALSE`, the output matrix column associated with codon '`AAA`' will contain `TRUE` for all codons that are non-synonymous to '`AAA`' and that differ from '`AAA`' at not more than two nucleotide positions. Codons that do not satisfy that criterion will be assigned `FALSE`. The same interpretation applies to the respective rows of the output matrix corresponding to the other sense codons. The output matrix is

symmetric, so all the interpretations hold similarly row-wise. The rows and columns of the output matrix are arranged and labelled alphabetically with respect to the IUPAC nucleotide triplet naming style.

Value

A 61-by-61 logical matrix. See **Details**.

Author(s)

Hassan Sadiq

See Also

Explanations that accompany the `diffNucBinary` function might provide better operational understanding.

Examples

```
testing <- nonSynonymous(FALSE, 2)
head(testing)[,seq(1,6)]
```

---

queeems *Quantify the Extent of Evolutionary Evidence in Molecular Sequences*

Description

Saturation indices are often used, in molecular evolutionary studies, to determine whether the degree of genetic variation captured in a set of genetic sequences is likely to produce reliable inferences. Existing measures are largely designed to cater for analyses about phylogeny reconstruction. As a result, a single metric is usually obtained for an alignment of homologous sequences. When saturation is detected, it is common practice to exclude sequence positions believed to significantly influence the erosion of genetic variation. The suite of functions presented here are designed to facilitate a more informed exclusion technique by providing Bayesian saturation metric for each sequence position.

Details

Bayes factors that measure the extent to which the recorded genetic data support an alternative hypothesis of saturation is estimated for each sequence position, in terms of codon bases. This approach avails an exciting opportunity for saturation indices to find use in other molecular evolutionary topics, including analyses of natural selection. Other independently useful functions are also part of the package, these include various molecular entropy quantifying functions.

Author(s)

Hassan Sadiq

References

Kass, R. E. and Raftery, A. E. (1995). Bayes Factors, *Journal of American Statistical Association* **90**(430): 773-795.

Renyi, A. (1961). On Measures of Information and Entropy, *Proceedings of the Fourth Berkeley Symposium on Mathematics, Statistics and Probability 1960* 547-561.

Shannon, C. E. (1948). A Mathematical Theory of Communication, *The Bell System Technical Journal* **27**(3): 379-423.

Xia, X., Xie, Z., Salemi, M., Chen, L. and Wang, Y. (2003). An Index of Substitution Saturation and its Application, *Molecular Phylogenetics and Evolution* **26**(1): 1-7.

`queeemsExtdata` *Paths to Example Protein Sequence Files*

Description

Obtain the paths to example protein sequences that are available, as part of, and are used in some of the examples in, the `queeems` package.

Usage

```
queeemsExtdata(filename=NULL)
```

Arguments

`filename`	A string of the name of one of the `.fasta` files packaged with `queeems`.

Details

The external data sets accessible with this function include the, `II.fasta`, `PI.fasta` and `RTI.fasta` DNA files as well as their amino acid translates. That is, `II_AA.fasta`, `PI_AA.fasta` and `RTI_AA.fasta`. The FASTA format files all contain empirical HIV-1 protein sequences that were retrieved from Murrell et al. (2012). Other files accessible from the folder include the `VertCOI.fasta`: mitochondrial COI sequences from eight vertebrates (Xia, 2018) and `InvertebrateEF1a.fasta`: elongation factor-1a sequences from chelicerate, myriapod, branchipod, hexapod, molluscan, annelid and malacostracan species. The `filename` input may be specified as any of the listed filnames or left as the default `NULL` value.

The reverse transcriptase isolates [`RTI.fasta`] contain 205 codon sites and 476 sequences. Following Murrell et al. (2012), this corresponds to a subset [codon sites 41 to 245] of the original (Rhee et al, 2003, de Oliveira et al,, 2010) data. The protease isolate [`PI.fasta`] file contains 99 codon sites and 122 protein sequences. The integrase isolates [`II.fasta`] comprise 295 sequences and 288 codon sites. All the cited HIV-1 data sets contain pre- and post-treatment sequences.

Value

A string that corresponds to the full path of the data saved with the specified `filename` input. If `filename` is left as its default `NULL` value, a list of all the names of all the sequence files that are included with the `queeems` package are listed.

Author(s)

Hassan Sadiq

References

Rhee, S.-Y., Gonzales, M. J., Kantor, R., et al. (2003). Human Immunodeficiency Virus Reverse Transcriptase and Protease Sequence Database, *Nucleic Acids Research* **31**(1):298-303.

de Oliveira, T., Shafer, R. W., Seebregts, C., et al. (2010). Public Database for HIV Drug Resistance in Southern Africa, *Nature* **464**(7289):673.

Murrell, B., de Oliveira, T., Seebregts, C., et al. (2012). Modeling HIV-1 Drug Resistance as Episodic Directional Selection, *PLoS Computational Biology* **8**(5):e1002507.

Xia, X. (2018). DAMBE7: New and Improved Tools for Data Analysis in Molecular Biology and Evolution, *Molecular Biology and Evolution* **35**(6):1550-1552.

See Also

See applications of this function as part of `bstringCodons`.

Examples

```
test0 <- queeemsExtdata()
print(test0)

test1 <- queeemsExtdata("II.fasta")
print(test1)
```

`saturateBF-class` *Substitution Saturation Output: Bayesian Paradigm*

Description

Allow for neat and efficient handling of outputs from Bayesian site-wise substitution saturation analyses.

Objects from the Class

Creating `saturateBF` objects may be achieved with calls stated as follows: `new("saturateBF", bf.pv, nuLL, altLL, noInfo, nsites)`.

Slots

`bf.pv`**:** numeric vector of the Bayes factors estimated independently for each molecular data site.

`nuLL`**:** log-likelihood recorded for each data site based on the null parameters of the corresponding saturation analyses.

`altLL`**:** estimates of the log-likelihood obtained for each data site, using the parameters of the alternative hypothesis.

`noInfo`**:** site indices of the analysed protein data sites that are observed to be invariant.

`nsites`**:** number of base sites that make up the original molecular data.

`bfcutoff`**:** Bayes factor threshold for determining significance.

`sspi`**:** tolerated proportion of saturated sites to use for overall hypotheses statements.

Methods

show: `signature(object="saturateBF")` self-explanatory output highlights.

summary: `signature(object="saturateBF")` interesting Bayesian-informed saturation statistics in properly labelled data frame.

nonvaries: `signature(cNS="saturateBF")` vector that contains indices of the sites in the analysed protein data that were invariant.

sitecount: `signature(gent="saturateBF")` number of base locations that makeup the molecular data used for the saturation analyses.

BFs: `signature(satube="saturateBF")` Bayes factor estimates obtained for each data site.

LogL0: `signature(satube="saturateBF")` estimates of the log-likelihood computed using the parameters of the null hypothesis, for each base site separately.

LogL1: `signature(satube="saturateBF")` log-likelihood estimates for each data site as obtained from application of the parameters of the alternative hypothesis.

mainPi: `signature(satube="saturateBF")` a positive fraction that specifies the value to use when conducting hypotheses tests for overall saturation.

BFthreshold: `signature(satube="saturateBF")` Bayes factor threshold value that determines when to infer saturation.

Author(s)

Hassan Sadiq

See Also

The `seqSaturation` function that produces objects of this class.

Examples

```
geneFile <- queeemsExtdata("II.fasta")
examine <- seqSaturation(geneFile, "A", 5, 0.4, 5)
examine
```

`seqfilter` *Delete Subset of Molecular Sequences*

Description

Extract targetted base sites from molecular sequence data.

Usage

```
seqfilter(fastafile, deadsites, basename="dna", write=FALSE)
```

Arguments

`fastafile`	full path to the file that contains the sequence that needs to be subsetted.
`deadsites`	integer indices of the base locations that should be removed from the sequence in `fastafile`.
`basename`	one of `` `dna` `` (default), `` `codon` `` or `` `aa` `` that indicates how the sequence data should be captured.
`write`	optional full path to a file where the output sequence should be saved.

Details

If entered, `write` should be the full path to the file where the data is expected to be saved. The input is not always interpreted as logical. If `write` = `FALSE` is entered (default), no file will be saved. However, `write` = `TRUE` will save the edited sequence output in a file named `` `TRUE` `` in the working directory.

Value

The extracted molecular sequence data as a `Biostrings::BStringSet` object. If `write` is specified, the edited sequence is also saved in the filepath stated.

Author(s)

Hassan Sadiq

Examples

```
egdata <- c("CCTCAGATCACTCTTTGGCAACGACCCCTT",
            "CCTCAGATCACTCTTGTCTCAATAAAAGTA",
            "TGG-AGCGACCCCTTGTCTCAATAAAAATA")
dnaseqs <- Biostrings::DNAStringSet(egdata)
names(dnaseqs) <- paste("seq", seq(1,3), sep="")
egpath <- file.path(tempdir(), "egdata.fas")
Biostrings::writeXStringSet(dnaseqs, egpath)
cleanseqs <- seqfilter(egpath, c(4,6,8), "dna")
print(cleanseqs)
```

`seqSaturation` *Estimate Saturation Index For Protein Data*

Description

Assess the degree of evolutionary saturation in molecular sequences.

Usage

```
seqSaturation(fastafile, approach="A", wsize=6, wless1=0.40,
    weightmax=7, bfcutoff=3, sspi=0.01)
```

Arguments

`fastafile`	a character input of the full path to a `.fasta` file that contains the genetic data to be analysed.
`approach`	one of 'A' (default), 'B' or 'C' that indicates the preferred saturation assessment method.
`wsize`	number of π parameter weight classes (default = 6).
`wless1`	fraction of π weight classes to be less than one (default = 0.40).
`weightmax`	maximum π weight value (default = 7).
`bfcutoff`	Bayes factor cutoff (default = 3).
`sspi`	tolerated proportion of saturated sites, overall.

Details

At least, three molecular saturation verification techniques are planned to be incorporated into the package. However, only one is currently accessible through the `seqSaturation` function. The logic behind the technique is presented below.

- **Technique 1 (`approach = "A"`):**
  Codons are combinations of nucleotide bases. Codons are therefore more appealing for rigorous evolutionary analyses, including the popular investigation of natural selection as a ratio of non-synonymous to synonymous codon substitution rates. In addition, due to the smaller set of nucleotide bases (= 4) compared to codon bases (= 64), the former are expected to lose evolutionary information faster. The substitution saturation test built upon this reasoning reduces to the following hypotheses statements.
  - *Null hypothesis (`H0`)*: Evolutionary information contained in codon data is significantly **more** than the information retrievable from associated nucleotide data. That is, sequence is **not** saturated. ($\texttt{Bayes Factor} \leq \texttt{BF}_{\texttt{cutoff}}$).
  - *Alternative hypothesis (`H1`)*: Evolutionary information retrievable in the codon data is significantly eroded or insufficient for evolutionary analyses, such that nucleotide models reveal more about the data. That is, sequence is saturated. ($\texttt{Bayes Factor} > \texttt{BF}_{\texttt{cutoff}}$). $\texttt{BF}_{\texttt{cutoff}}$ is the input `bfcutoff`. To verify these hypotheses, it is assumed that the number of codon and nucleotide bases observed at each of the corresponding positions of

genetic data follows a multinomial distribution. Bayes factors are then estimated independently for each codon site, `i = (1, 2, ..., n)`, as the ratio of maximum posterior log-likelihoods as follows.

$$\mathsf{BF}_i = \frac{LogL\left(X_{\star i}^{(k)}|\boldsymbol{\pi}_{\star i}^{(k)}\right)}{LogL\left(Y_{\star i}|\boldsymbol{\pi}_{\star i}^{\mathsf{cdn}}\right)} \propto \frac{\sum_{k=1}^{3}\left(\max\limits_{\boldsymbol{\pi}^{(k)}}\sum_{j=1}^{4}x_{ji}^{(k)} \times \log\left(\pi_{ji}^{(k)}\right)\right)}{\max\limits_{\boldsymbol{\pi}^{\mathsf{cdn}}}\sum_{m=1}^{61}y_{mi} \times \log\left(\pi_{mi}^{\mathsf{cdn}}\right)},$$

where,

* $Y_{\star i}$ = data at codon position `i`,
* $\boldsymbol{\pi}_{\star i}^{\mathsf{cdn}}$ = theoretical proportions of sense codons at site `i`,
* $X_{\star i}^{(k)}$ = nucleotide data in position `k = (1, 2, 3)` at codon site `i`,
* $\boldsymbol{\pi}_{\star i}^{(k)}$ = theoretical proportions of nucleotide bases at position `k` of codon site `i`,
* y_{mi} = number of sense codon `m` observed at codon `i`,
* $x_{ji}^{(k)}$ = number of nucleotide base `j` recorded at position `k` of codon site `i`.

Values of $\boldsymbol{\pi}$ that the likelihoods are maximised over for each site are pre-determined based on a technique adopted from FUBAR. Weights of size `w = nweights` ≥ 4 within the `(0, weightmax)` interval are first generated such that `wless1` fraction of the weights are less than one. For more details about the weight generation process, see `fubarweights`. All possible, $z = (w!)/(w-4)!$, four-weight combinations without repetitions are then sampled and converted to `z` vectors of nucleotide frequencies $\left(\boldsymbol{\pi}^{(k)}\right)$ using the `softmax` transformation. The z codon frequencies $\left(\boldsymbol{\pi}^{\mathsf{cdn}}\right)$ considered at each site are obtained as the product of the frequencies of the associated nucleotide triplets as it is conventionally implemented in the `F3x4` substitution models.

The log-likelihood expressions are similar to the popular Shannon information entropy expression. The similarity justifies why it is not inappropriate to have constructed the hypotheses statements in terms of the information that can be retrieved. Note that the log-lilihood estimates will always be negative for both the numerator and denominator of the Bayes factor formula, because the `x, y` counts are strictly non-negative and the theoretical proportions (π) are always fractions. Consequently, to avoid spurious outcomes, the log-likelihood estimates were transformed into approximate posterior probabilities using the `softmax` identity before maximising.

A single Bayes factor is also estimated to assess the overall extent of saturation of the genetic sequence data (Y). Separate hypotheses statements about the true proportion of saturated sites (ρ), were necessary for this purpose. These hypotheses may be expressed as follows.

$$\mathsf{H}_0 : \rho \leq \rho_0 \mathsf{H}_1 : \rho > \rho_0,$$

where ρ is the parameter of the `Binomial(n, ρ)` distribution, `n` = the number of codon sites and ρ_0 is the input `sspi` (default `0.10`). Similar to the approach taken for the site-wise inferences, potential $\rho \in (0, 1)$ values are obtained using the FUBAR discretisation method. A Bayes factor is then estimated as the ratio of the maximum posterior probabilities as follows.

$$\mathsf{BF}_{\mathtt{overall}} = \frac{L\left(Y\middle|\rho > \rho_0\right)}{L\left(Y\middle|\rho \leq \rho_0\right)}.$$

- **Technique 2** (`approach = "B"`) (in progress):
  A significant number of existing codon models of evolution tolerate only one nucleotide mismatch between codon pairs. The assumption is that not enough time has passed for more than a single substitution event to have occurred. The following hypotheses statements are plausible consequence of this widely adopted assumption.
  - *Null hypothesis (`H0`)*: The frequencies of the codon bases that make up the genetic data are better described by a model that supports only a single nucleotide mismatch between codon pairs. That is, sequence is **not** saturated. (`Bayes Factor` $\leq$ $\texttt{BF}_{\texttt{cutoff}}$).
  - *Alternative hypothesis (`H1`)*: Observed base frequencies distribution is better supported by a model that permits both two and three nucleotide mismatches per codon pair. That is, sequence is saturated (`Bayes Factor` $>$ $\texttt{BF}_{\texttt{cutoff}}$).
- **Technique 3** (`approach = "C"`) (in progress):
  This should extend `Technique 2` by allowing for the hypotheses to be nested such that Chi-Square likelihood ratio test becomes applicable with the following updated hypothesis statements.
  - *Null hypothesis (`H0`)*: Considering only one nucleotide mismatch per codon pair captures majority of the evidence of evolution that has accrued in the genetic data. That is, sequence is **not** saturated.
  - *Alternative hypothesis (`H1`)*: Including information about one, two and three nucleotide mismatches per codon pair captures significantly more evolutionary evidence from the genetic data. That is, sequence is saturated.

Value

The output object returned is determined by the analyses technique. Additional techniques should be incorporated soon. Bayesian outputs from `Technique 1` is returned as a `saturateBF` object. The object contains details of the site-wise likelihood estimates and other useful characteristic information. The respective object help file may be consulted for further details.

Author(s)

Hassan Sadiq

References

Benjamini, Y. and Hochberg, Y. (1995). Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing, *Journal of the Royal Statistical Society: Series B*. **57**(1): 289-300.

Efron, B. and Hastie, T. (2016). *Computer Age Statistical Inference. Algorithms, Evidence and Data Science*. Cambridge, UK: Cambridge University Press.

Goldman, N. and Yang, Z. (1994) A Codon-Based Model of Nucleotide Substitution for Protein-Coding DNA Sequences, *Molecular Biology and Evolution* **11**(5): 725-736.

Kass, R. E. and Raftery, A. E. (1995) Bayes Factors, *Journal of the American Statistical Association* **90**(430): 773-795.

Murrell, B., Moola, S., Mabona, A., Weighill, T., Sheward, D., Kosakovsky Pond, S. L. and Scheffler, K. (2013) FUBAR: A Fast, Unconstrained Bayesian AppRoximation for Inferring Selection, *Molecular Biology and Evolution* **30**(5): 1196-1205.

Renyi, A. (1961). On Measures of Information and Entropy, *Proceedings of the Fourth Berkeley Symposium on Mathematics, Statistics and Probability 1960* 547-561.

Shannon, C. E. (1948). A Mathematical Theory of Communication, *The Bell System Technical Journal* **27**(3): 379-423.

Xia, X., Xie, Z., Salemi, M., Chen, L. and Wang, Y. (2003). An Index of Substitution Saturation and its Application, *Molecular Phylogenetics and Evolution* **26**(1): 1-7.

See Also

- Molecular entropy estimation functions:
  - `entropyindex`
  - `codondifferindex`
  - `entropyindex`
  - `molentropy`
- Multinomial likelihood generating function:
  - `mnomLogl`
- Functions useful for transforming molecular data into the `x, y` counts used in the equations above:
  - `baseFrequency`
  - `codonDissimilarity`
- Weight generating function.
  - `fubarweights`
- Function that converts nucleotide to codon frequencies.
  - `n2cFreqs`
- Softmax identity function.
  - `softmax`

Examples

```
proteinFile <- queeemsExtdata("VertCOI.fasta")
testEG <- seqSaturation(proteinFile, "A", 5, 0.4, 5)
summary(testEG)
testEG
```

`siteindices-class` *Molecular Information Entropy Processor*

Description

Allow for neat and efficient handling of information entropy indices outputs generated site-wise from molecular data.

Objects from the Class

To create a `siteindices` object require executing a call of the form `new("siteindices", eindices, noinfo, meth, alphaR)`.

Slots

`eindices`**:** vector of site-wise entropy estimates.

`noinfo`**:** numeric vector that contains positions of the molecular data that produced `eindices` which were non-informative.

`meth`**:** character entry that informs about the method used to generate `eindices` from the original protein data.

`alphaR`**:** parameter for Renyi entropy calculations. Only relevant if `meth = "renyi"`.

Methods

show: `signature(object="siteindices")` output highlights.

summary: `signature(object="siteindices")` interesting output statistics.

nonvaries: `signature(cNS="siteindices")` positions of the processed molecular data that are not informative.

renyiA: `signature(sent="siteindices")` value of the alpha parameter used for Renyi entropy calculations.

informula: `signature(sent="siteindices")` should return the name of the method used to generate the entropy estimates from the original molecular data.

sitentropies: `signature(sent="siteindices")` values of entropy estimates obtained for each site of a genetic sequence.

Details

An approximate 95% confidence interval estimated using the Bienayme-Tchebcshef inequality is returned as part of the outputs of the `siteindices` class. The expression for the said confidence interval calculation is as follows.

$$\bar{H} \pm k \times S_H/\sqrt{n},$$

where $\bar{H}$ and S_H are the average and the standard deviation of the n entropy estimates supplied as the input for the `eindices` slot. The parameter $k = \sqrt{1/0.05}$.

Author(s)

Hassan Sadiq

References

Feller, W. (1968). An Introduction to Probability Theory and its Applications: Vol. 1, *Wiley New York*.

See Also

- Variant of the nonvaries method as applied within the baseSummary and the geneindex classes.
- The entropyindex function is a good reference for discussions about the different entropy calculation techniques accessible from the queeems package.
- Functions that return outputs of the described siteindices-class include codondifferindex and molentropy.

Examples

```
proteinFile <- queeemsExtdata("VertCOI.fasta")
testEG <- codondifferindex(proteinFile, 1, "shannon")
sitentropies(testEG)[seq(1,10)]
nonvaries(testEG)[seq(1,10)]
summary(testEG)
testEG
```

---

softmax *Softmax Transformation*

Description

Transform weights to frequencies using the softmax identity

Usage

```
softmax(weights, maxadj)
```

Arguments

weights	vector of weights.
maxadj	logical indicator of whether weights need adjustment.

Details

The softmax identity is popular in machine learning applications. It is useful for transforming a vector of weights into frequencies. A modified version that minimises the chances of overflow is applied here. For a given vector (x = weights) that contains k real number weights, the ith frequency (where, i = 1, 2, ..., k) is obtained as follows.

$$\pi_i = \frac{e^{x_i - c}}{\sum_{j=1}^{k} e^{x_j - c}},$$

where c is set equal to zero if maxadj = FALSE. If maxadj = TRUE then, $c = \max_j x_j$.

Value

A numeric vector of frequencies that sum to one.

Author(s)

Hassan Sadiq

References

Efron, B. and Hastie, T. (2016). *Computer Age Statistical Inference. Algorithms, Evidence and Data Science*. Cambridge, UK: Cambridge University Press.

Examples

```
wData <- sample(seq(1,6), 20, replace=TRUE)
test1 <- softmax(wData, FALSE)
test2 <- softmax(wData, TRUE)
print(test1)
print(test2)
```

Index

∗ **classes**
  baseSummary-class, 4
  citing-class, 6
  cncsframe-class, 10
  geneindex-class, 16
  saturateBF-class, 24
  siteindices-class, 30

aboutQueeems, 2, *6*

basecensus *(*baseSummary-class*)*, 4
basecensus,baseSummary-method *(*baseSummary-class*)*, 4
baseFrequency, 3, *5*, *13*, *15*, *19*, *20*, *30*
baseSummary, *4*, *17*, *32*
baseSummary *(*baseSummary-class*)*, 4
baseSummary-class, 4
BFs *(*saturateBF-class*)*, 24
BFs,saturateBF-method *(*saturateBF-class*)*, 24
BFthreshold *(*saturateBF-class*)*, 24
BFthreshold,saturateBF-method *(*saturateBF-class*)*, 24
bstringCodons, 5, *24*
bstringCodons,BStringSet-method *(*bstringCodons*)*, 5
bstringCodons.BStringSet *(*bstringCodons*)*, 5

citing *(*citing-class*)*, 6
citing-class, 6
CnCs, *4*, 7, *9–11*, *15*, *19*
cncsentropy, 8, *17*, *19*, *20*
cncsframe, *7*, *8*
cncsframe *(*cncsframe-class*)*, 10
cncsframe-class, 10
codondifferindex, *9*, 11, *20*, *30*, *32*
codonDissimilarity, *8*, *11*, 12, *15*, *19*, *30*
codonDissimilarity,BStringSet,numeric-method *(*codonDissimilarity*)*, 12
codonDissimilarity.BStringSet *(*codonDissimilarity*)*, 12

datatype *(*baseSummary-class*)*, 4
datatype,baseSummary-method *(*baseSummary-class*)*, 4
diffNucBinary, *4*, 13, *22*

entropyindex, *9*, *11*, 14, *17*, *20*, *30*, *32*

fubarweights, 15, *28*, *30*

geneindex, *9*, *32*
geneindex *(*geneindex-class*)*, 16
geneindex-class, 16
gentropy *(*geneindex-class*)*, 16
gentropy,geneindex-method *(*geneindex-class*)*, 16

II.fasta *(*queeemsExtdata*)*, 23
informula *(*geneindex-class*)*, 16
informula,geneindex-method *(*geneindex-class*)*, 16
informula,siteindices-method *(*siteindices-class*)*, 30

LogL0 *(*saturateBF-class*)*, 24
LogL0,saturateBF-method *(*saturateBF-class*)*, 24
LogL1 *(*saturateBF-class*)*, 24
LogL1,saturateBF-method *(*saturateBF-class*)*, 24

mainPi *(*saturateBF-class*)*, 24
mainPi,saturateBF-method *(*saturateBF-class*)*, 24
mnomLogl, *15*, 18, *30*
molentropy, *9*, *11*, 19, *30*, *32*

n2cFreqs, 20, *30*
nonSynonymous, *8*, 21

nonvaries (cncsframe-class), 10
nonvaries,cncsframe-method (cncsframe-class), 10
nonvaries,geneindex-method (geneindex-class), 16
nonvaries,saturateBF-method (saturateBF-class), 24
nonvaries,siteindices-method (siteindices-class), 30
nORs (cncsframe-class), 10
nORs,cncsframe-method (cncsframe-class), 10
nseqs (baseSummary-class), 4
nseqs,baseSummary-method (baseSummary-class), 4
nsfreqs (cncsframe-class), 10
nsfreqs,cncsframe-method (cncsframe-class), 10
nucbalance (cncsframe-class), 10
nucbalance,cncsframe-method (cncsframe-class), 10

PI.fasta (queeemsExtdata), 23

queeems, *11*, *17*, *19*, 22, *32*
queeems-package (queeems), 22
queeemsExtdata, *4*, *13*, 23

renyiA (geneindex-class), 16
renyiA,geneindex-method (geneindex-class), 16
renyiA,siteindices-method (siteindices-class), 30
RTI.fasta (queeemsExtdata), 23

saturateBF, *29*
saturateBF (saturateBF-class), 24
saturateBF-class, 24
seqfilter, 26
seqSaturation, *15*, *19*, *25*, 27
show, citing-method (citing-class), 6
show,baseSummary-method (baseSummary-class), 4
show,cncsframe-method (cncsframe-class), 10
show,geneindex-method (geneindex-class), 16
show,saturateBF-method (saturateBF-class), 24
show,siteindices-method (siteindices-class), 30
si.renyi (entropyindex), 14
si.shannon (entropyindex), 14
sitecount (geneindex-class), 16
sitecount,geneindex-method (geneindex-class), 16
sitecount,saturateBF-method (saturateBF-class), 24
siteindices, *11*, *17*, *19*
siteindices (siteindices-class), 30
siteindices-class, 30
sitentropies (siteindices-class), 30
sitentropies,siteindices-method (siteindices-class), 30
softmax, *28*, *30*, 32
summary,saturateBF-method (saturateBF-class), 24
summary,siteindices-method (siteindices-class), 30

useSyn (geneindex-class), 16
useSyn,geneindex-method (geneindex-class), 16

VertCOI.fasta (queeemsExtdata), 23

wildtriplets (cncsframe-class), 10
wildtriplets,cncsframe-method (cncsframe-class), 10