

Package ‘biogram’

March 31, 2020

Type Package

Title N-Gram Analysis of Biological Sequences

Version 1.6.3

LazyData true

Date 2020-03-31

Description Tools for extraction and analysis of various n-grams (k-mers) derived from biological sequences (proteins or nucleic acids). Contains QuiPT (quick permutation test) for fast feature-filtering of the n-gram data.

License GPL-3

URL <https://github.com/michbur/biogram>

BugReports <https://github.com/michbur/biogram/issues>

VignetteBuilder knitr

Depends R (>= 3.0.0), slam

Imports combinat, entropy, partitions

Suggests ggplot2, knitr, testthat

NeedsCompilation no

Repository CRAN

Encoding UTF-8

RoxygenNote 7.1.0

Author Michal Burdukiewicz [cre, aut]
(<<https://orcid.org/0000-0001-8926-582X>>),
Piotr Sobczyk [aut],
Chris Lauber [aut],
Dominik Rafacz [aut],
Katarzyna Sidorcuk [ctb] (<<https://orcid.org/0000-0001-6576-9054>>)

Maintainer Michal Burdukiewicz <michalburdukiewicz@gmail.com>

Date/Publication 2020-03-31 14:30:06 UTC

R topics documented:

biogram-package	3
aaprop	5
add_1grams	21
as.data.frame.feature_test	22
binarize	23
calc_criterion	23
calc_cs	24
calc_ed	25
calc_ig	26
calc_kl	27
calc_pi	28
calc_si	29
check_criterion	30
cluster_reg_exp	31
code_ngrams	32
construct_ngrams	32
count_multigrams	34
count_ngrams	35
count_specified	37
count_total	38
create_encoding	39
create_feature_target	40
create_ngrams	40
criterion_distribution	41
cut.feature_test	42
decode_ngrams	42
degenerate	43
degenerate_ngrams	44
distr_crit	45
encoding2df	46
fast_crosstable	46
feature_test	47
full2simple	48
gap_ngrams	49
generate_sequence	49
generate_single_region	50
generate_single_unigram	51
generate_unigrams	51
get_ngrams_ind	52
human_cleave	53
is_ngram	54
l2n	54
list2matrix	55
n2l	56
ngrams2df	56
plot.criterion_distribution	57

position_ngrams	58
print.feature_test	59
read_fasta	59
regenerate	60
regional_param	61
seq2ngrams	61
simple2full	62
summary.feature_test	63
table_ngrams	63
test_features	64
validate_encoding	66
write_encoding	67
write_fasta	68

Index	69
--------------	-----------

biogram-package	<i>biogram - analysis of biological sequences using n-grams</i>
-----------------	---

Description

biogram package is a toolbox for the analysis of nucleic acid and protein sequences using n-grams. Possible applications include motif discovery, feature selection, clustering, and classification.

n-grams

n-grams (k-tuples) are sets of n characters derived from the input sequence(s). They may form continuous sub-sequences or be discontinuous. For example, from the sequence of nucleotides AATA one can extract the following continuous 2-grams (bigrams): AA, AT and TA. Moreover, there are two possible bigrams separated by a single space: A_T and A_A, and one bigram separated by two spaces: A__A.

Another important n-gram parameter is its position. Instead of just counting n-grams, one may want to count how many n-grams occur at a given position in multiple (e.g. related) sequences. For example, in the sequences AATA and AACA there is only one bigram at position 1: AA, but there are two bigrams at position two: AT and AC. The following notation is used for position-specific n-grams: 1_AA, 2_AT, 2_AC.

In the biogram package, the `count_ngrams` function is used for counting and extracting n-grams. Using the `d` argument the user can specify the distance between elements of the n-grams. The `pos` argument can be used to enable position specificity.

n-gram data dimensionality

We note that n-grams suffer from the curse of dimensionality. For example, for a peptide of length 6 20^n n-grams and 6×20^n positioned n-grams are possible. Data sets of such an enormous size are hard to manage and analyze in R.

The biogram package deals with both of the abovementioned problems. It uses innate properties of the n-gram data which usually can be represented by sparse matrices. Data storage is done

using functionalities from the `slam` package. To ease the selection of significant features, `biogram` provides the user with QuiPT, a very fast permutation test for binary data (see [test_features](#)).

Another way of reducing dimensionality is the aggregation of sequence residues into more general groups. For example, all positively-charged amino acids may be aggregated into one group. This action can be performed using the `degenerate` function.

Encoding of amino acids can ease sequence analysis, but multidimensional objects as the aggregations of amino acids are not easily comparable. We introduced the encoding distance, a measure defining the distance between encodings. It can be computed using the `calc_ed` function.

Author(s)

Michal Burdukiewicz, Piotr Sobczyk, Chris Lauber

Examples

```
# use data set from package
data(human_cleave)
# first nine columns represent subsequent nine amino acids from cleavage sites
# degenerate the sequence to reduce the dimensionality of the problem
# (use five groups instead of 20 amino acids)
deg_seqs <- degenerate(human_cleave[, 1L:9],
                      list(`a` = c(1, 6, 8, 10, 11, 18),
                           `b` = c(2, 13, 14, 16, 17),
                           `c` = c(5, 19, 20),
                           `d` = c(7, 9, 12, 15),
                           `e` = c(3, 4)))

# EXAMPLE 1 - extract significant trigrams
# extract trigrams
trigrams <- count_ngrams(deg_seqs, 3, letters[1L:5], pos = TRUE)
# select features that differ between the two target groups using QuiPT
test1 <- test_features(human_cleave[, "tar"], trigrams)
# see a summary of the results
summary(test1)
# aggregate features in groups based on their p-value
gr <- cut(test1)
# get position map of the most significant n-grams
position_ngrams(gr[[1]])
# transform the most significant n-grams to more readable form
decode_ngrams(gr[[1]])

# EXAMPLE 2 - search for specific n-grams
# the n-grams of the interest are a_a (a-gap-a) and e_e (e-gap-e) on the
# 3rd and 4th position
# firstly code n-grams in biogram notation and add position information
coded <- code_ngrams(c("a_a", "c_c"))
# add position information
coded <- c(paste0("3_", coded), paste0("4_", coded))
# count only the features of the interest
bigrams <- count_specified(deg_seqs, coded)
# test which of the features of the interest is significant
test2 <- test_features(human_cleave[, "tar"], bigrams)
```

```
cut(test2)
```

aaprop

Normalized amino acids properties

Description

Normalized (0-1) 554 amino acid properties as retrieved from AAIndex database (release 9.1) enriched with contactivity of amino acids.

Format

A data frames with 20 columns and 600 rows.

Details

Following properties are included (AAIndex key: description of the property)

- ANDN920101** alpha-CH chemical shifts (Andersen et al., 1992)
- ARGP820101** Hydrophobicity index (Argos et al., 1982)
- ARGP820102** Signal sequence helical potential (Argos et al., 1982)
- ARGP820103** Membrane-buried preference parameters (Argos et al., 1982)
- BEGF750101** Conformational parameter of inner helix (Beghin-Dirkx, 1975)
- BEGF750102** Conformational parameter of beta-structure (Beghin-Dirkx, 1975)
- BEGF750103** Conformational parameter of beta-turn (Beghin-Dirkx, 1975)
- BHAR880101** Average flexibility indices (Bhaskaran-Ponnuswamy, 1988)
- BIGC670101** Residue volume (Bigelow, 1967)
- BIOV880101** Information value for accessibility; average fraction 35% (Biou et al., 1988)
- BIOV880102** Information value for accessibility; average fraction 23% (Biou et al., 1988)
- BROC820101** Retention coefficient in TFA (Browne et al., 1982)
- BROC820102** Retention coefficient in HFBA (Browne et al., 1982)
- BULH740101** Transfer free energy to surface (Bull-Breese, 1974)
- BULH740102** Apparent partial specific volume (Bull-Breese, 1974)
- BUNA790101** alpha-NH chemical shifts (Bundi-Wuthrich, 1979)
- BUNA790102** alpha-CH chemical shifts (Bundi-Wuthrich, 1979)
- BUNA790103** Spin-spin coupling constants $3J_{\text{H}\alpha\text{-NH}}$ (Bundi-Wuthrich, 1979)
- BURA740101** Normalized frequency of alpha-helix (Burgess et al., 1974)
- BURA740102** Normalized frequency of extended structure (Burgess et al., 1974)
- CHAM810101** Steric parameter (Charton, 1981)
- CHAM820101** Polarizability parameter (Charton-Charton, 1982)
- CHAM820102** Free energy of solution in water, kcal/mole (Charton-Charton, 1982)

- CHAM830101** The Chou-Fasman parameter of the coil conformation (Charton-Charton, 1983)
- CHAM830102** A parameter defined from the residuals obtained from the best correlation of the Chou-Fasman parameter of beta-sheet (Charton-Charton, 1983)
- CHAM830103** The number of atoms in the side chain labelled 1+1 (Charton-Charton, 1983)
- CHAM830104** The number of atoms in the side chain labelled 2+1 (Charton-Charton, 1983)
- CHAM830105** The number of atoms in the side chain labelled 3+1 (Charton-Charton, 1983)
- CHAM830106** The number of bonds in the longest chain (Charton-Charton, 1983)
- CHAM830107** A parameter of charge transfer capability (Charton-Charton, 1983)
- CHAM830108** A parameter of charge transfer donor capability (Charton-Charton, 1983)
- CHOC750101** Average volume of buried residue (Chothia, 1975)
- CHOC760101** Residue accessible surface area in tripeptide (Chothia, 1976)
- CHOC760102** Residue accessible surface area in folded protein (Chothia, 1976)
- CHOC760103** Proportion of residues 95% buried (Chothia, 1976)
- CHOC760104** Proportion of residues 100% buried (Chothia, 1976)
- CHOP780101** Normalized frequency of beta-turn (Chou-Fasman, 1978a)
- CHOP780201** Normalized frequency of alpha-helix (Chou-Fasman, 1978b)
- CHOP780202** Normalized frequency of beta-sheet (Chou-Fasman, 1978b)
- CHOP780203** Normalized frequency of beta-turn (Chou-Fasman, 1978b)
- CHOP780204** Normalized frequency of N-terminal helix (Chou-Fasman, 1978b)
- CHOP780205** Normalized frequency of C-terminal helix (Chou-Fasman, 1978b)
- CHOP780206** Normalized frequency of N-terminal non helical region (Chou-Fasman, 1978b)
- CHOP780207** Normalized frequency of C-terminal non helical region (Chou-Fasman, 1978b)
- CHOP780208** Normalized frequency of N-terminal beta-sheet (Chou-Fasman, 1978b)
- CHOP780209** Normalized frequency of C-terminal beta-sheet (Chou-Fasman, 1978b)
- CHOP780210** Normalized frequency of N-terminal non beta region (Chou-Fasman, 1978b)
- CHOP780211** Normalized frequency of C-terminal non beta region (Chou-Fasman, 1978b)
- CHOP780212** Frequency of the 1st residue in turn (Chou-Fasman, 1978b)
- CHOP780213** Frequency of the 2nd residue in turn (Chou-Fasman, 1978b)
- CHOP780214** Frequency of the 3rd residue in turn (Chou-Fasman, 1978b)
- CHOP780215** Frequency of the 4th residue in turn (Chou-Fasman, 1978b)
- CHOP780216** Normalized frequency of the 2nd and 3rd residues in turn (Chou-Fasman, 1978b)
- CIDH920101** Normalized hydrophobicity scales for alpha-proteins (Cid et al., 1992)
- CIDH920102** Normalized hydrophobicity scales for beta-proteins (Cid et al., 1992)
- CIDH920103** Normalized hydrophobicity scales for alpha+beta-proteins (Cid et al., 1992)
- CIDH920104** Normalized hydrophobicity scales for alpha/beta-proteins (Cid et al., 1992)
- CIDH920105** Normalized average hydrophobicity scales (Cid et al., 1992)
- COHE430101** Partial specific volume (Cohn-Edsall, 1943)

CRAJ730101 Normalized frequency of middle helix (Crawford et al., 1973)
CRAJ730102 Normalized frequency of beta-sheet (Crawford et al., 1973)
CRAJ730103 Normalized frequency of turn (Crawford et al., 1973)
DAWD720101 Size (Dawson, 1972)
DAYM780101 Amino acid composition (Dayhoff et al., 1978a)
DAYM780201 Relative mutability (Dayhoff et al., 1978b)
DESM900101 Membrane preference for cytochrome b: MPH89 (Degli Esposti et al., 1990)
DESM900102 Average membrane preference: AMP07 (Degli Esposti et al., 1990)
EISD840101 Consensus normalized hydrophobicity scale (Eisenberg, 1984)
EISD860101 Solvation free energy (Eisenberg-McLachlan, 1986)
EISD860102 Atom-based hydrophobic moment (Eisenberg-McLachlan, 1986)
EISD860103 Direction of hydrophobic moment (Eisenberg-McLachlan, 1986)
FASG760101 Molecular weight (Fasman, 1976)
FASG760102 Melting point (Fasman, 1976)
FASG760103 Optical rotation (Fasman, 1976)
FASG760104 pK-N (Fasman, 1976)
FASG760105 pK-C (Fasman, 1976)
FAUJ830101 Hydrophobic parameter pi (Fauchere-Pliska, 1983)
FAUJ880101 Graph shape index (Fauchere et al., 1988)
FAUJ880102 Smoothed epsilon steric parameter (Fauchere et al., 1988)
FAUJ880103 Normalized van der Waals volume (Fauchere et al., 1988)
FAUJ880104 STERIMOL length of the side chain (Fauchere et al., 1988)
FAUJ880105 STERIMOL minimum width of the side chain (Fauchere et al., 1988)
FAUJ880106 STERIMOL maximum width of the side chain (Fauchere et al., 1988)
FAUJ880107 N.m.r. chemical shift of alpha-carbon (Fauchere et al., 1988)
FAUJ880108 Localized electrical effect (Fauchere et al., 1988)
FAUJ880109 Number of hydrogen bond donors (Fauchere et al., 1988)
FAUJ880110 Number of full nonbonding orbitals (Fauchere et al., 1988)
FAUJ880111 Positive charge (Fauchere et al., 1988)
FAUJ880112 Negative charge (Fauchere et al., 1988)
FAUJ880113 pK-a(RCOOH) (Fauchere et al., 1988)
FINA770101 Helix-coil equilibrium constant (Finkelstein-Ptitsyn, 1977)
FINA910101 Helix initiation parameter at position i-1 (Finkelstein et al., 1991)
FINA910102 Helix initiation parameter at position i,i+1,i+2 (Finkelstein et al., 1991)
FINA910103 Helix termination parameter at position j-2,j-1,j (Finkelstein et al., 1991)
FINA910104 Helix termination parameter at position j+1 (Finkelstein et al., 1991)
GARJ730101 Partition coefficient (Garel et al., 1973)

GEIM800101 Alpha-helix indices (Geisow-Roberts, 1980)
GEIM800102 Alpha-helix indices for alpha-proteins (Geisow-Roberts, 1980)
GEIM800103 Alpha-helix indices for beta-proteins (Geisow-Roberts, 1980)
GEIM800104 Alpha-helix indices for alpha/beta-proteins (Geisow-Roberts, 1980)
GEIM800105 Beta-strand indices (Geisow-Roberts, 1980)
GEIM800106 Beta-strand indices for beta-proteins (Geisow-Roberts, 1980)
GEIM800107 Beta-strand indices for alpha/beta-proteins (Geisow-Roberts, 1980)
GEIM800108 Aperiodic indices (Geisow-Roberts, 1980)
GEIM800109 Aperiodic indices for alpha-proteins (Geisow-Roberts, 1980)
GEIM800110 Aperiodic indices for beta-proteins (Geisow-Roberts, 1980)
GEIM800111 Aperiodic indices for alpha/beta-proteins (Geisow-Roberts, 1980)
GOLD730101 Hydrophobicity factor (Goldsack-Chalifoux, 1973)
GOLD730102 Residue volume (Goldsack-Chalifoux, 1973)
GRAR740101 Composition (Grantham, 1974)
GRAR740102 Polarity (Grantham, 1974)
GRAR740103 Volume (Grantham, 1974)
GUYH850101 Partition energy (Guy, 1985)
HOPA770101 Hydration number (Hopfinger, 1971), Cited by Charton-Charton (1982)
HOPT810101 Hydrophilicity value (Hopp-Woods, 1981)
HUTJ700101 Heat capacity (Hutchens, 1970)
HUTJ700102 Absolute entropy (Hutchens, 1970)
HUTJ700103 Entropy of formation (Hutchens, 1970)
ISOY800101 Normalized relative frequency of alpha-helix (Isogai et al., 1980)
ISOY800102 Normalized relative frequency of extended structure (Isogai et al., 1980)
ISOY800103 Normalized relative frequency of bend (Isogai et al., 1980)
ISOY800104 Normalized relative frequency of bend R (Isogai et al., 1980)
ISOY800105 Normalized relative frequency of bend S (Isogai et al., 1980)
ISOY800106 Normalized relative frequency of helix end (Isogai et al., 1980)
ISOY800107 Normalized relative frequency of double bend (Isogai et al., 1980)
ISOY800108 Normalized relative frequency of coil (Isogai et al., 1980)
JANJ780101 Average accessible surface area (Janin et al., 1978)
JANJ780102 Percentage of buried residues (Janin et al., 1978)
JANJ780103 Percentage of exposed residues (Janin et al., 1978)
JANJ790101 Ratio of buried and accessible molar fractions (Janin, 1979)
JANJ790102 Transfer free energy (Janin, 1979)
JOND750101 Hydrophobicity (Jones, 1975)
JOND750102 pK (-COOH) (Jones, 1975)

JOND920101 Relative frequency of occurrence (Jones et al., 1992)
JOND920102 Relative mutability (Jones et al., 1992)
JUKT750101 Amino acid distribution (Jukes et al., 1975)
JUNJ780101 Sequence frequency (Jungck, 1978)
KANM800101 Average relative probability of helix (Kanehisa-Tsong, 1980)
KANM800102 Average relative probability of beta-sheet (Kanehisa-Tsong, 1980)
KANM800103 Average relative probability of inner helix (Kanehisa-Tsong, 1980)
KANM800104 Average relative probability of inner beta-sheet (Kanehisa-Tsong, 1980)
KARP850101 Flexibility parameter for no rigid neighbors (Karplus-Schulz, 1985)
KARP850102 Flexibility parameter for one rigid neighbor (Karplus-Schulz, 1985)
KARP850103 Flexibility parameter for two rigid neighbors (Karplus-Schulz, 1985)
KHAG800101 The Kerr-constant increments (Khanarian-Moore, 1980)
KLEP840101 Net charge (Klein et al., 1984)
KRIW710101 Side chain interaction parameter (Krigbaum-Rubin, 1971)
KRIW790101 Side chain interaction parameter (Krigbaum-Komoriya, 1979)
KRIW790102 Fraction of site occupied by water (Krigbaum-Komoriya, 1979)
KRIW790103 Side chain volume (Krigbaum-Komoriya, 1979)
KYTJ820101 Hydropathy index (Kyte-Doolittle, 1982)
LAWE840101 Transfer free energy, CHP/water (Lawson et al., 1984)
LEVM760101 Hydrophobic parameter (Levitt, 1976)
LEVM760102 Distance between C-alpha and centroid of side chain (Levitt, 1976)
LEVM760103 Side chain angle theta(AAR) (Levitt, 1976)
LEVM760104 Side chain torsion angle phi(AAAR) (Levitt, 1976)
LEVM760105 Radius of gyration of side chain (Levitt, 1976)
LEVM760106 van der Waals parameter R0 (Levitt, 1976)
LEVM760107 van der Waals parameter epsilon (Levitt, 1976)
LEVM780101 Normalized frequency of alpha-helix, with weights (Levitt, 1978)
LEVM780102 Normalized frequency of beta-sheet, with weights (Levitt, 1978)
LEVM780103 Normalized frequency of reverse turn, with weights (Levitt, 1978)
LEVM780104 Normalized frequency of alpha-helix, unweighted (Levitt, 1978)
LEVM780105 Normalized frequency of beta-sheet, unweighted (Levitt, 1978)
LEVM780106 Normalized frequency of reverse turn, unweighted (Levitt, 1978)
LEWP710101 Frequency of occurrence in beta-bends (Lewis et al., 1971)
LIFS790101 Conformational preference for all beta-strands (Lifson-Sander, 1979)
LIFS790102 Conformational preference for parallel beta-strands (Lifson-Sander, 1979)
LIFS790103 Conformational preference for antiparallel beta-strands (Lifson-Sander, 1979)
MANP780101 Average surrounding hydrophobicity (Manavalan-Ponnuswamy, 1978)

- MAXF760101** Normalized frequency of alpha-helix (Maxfield-Scheraga, 1976)
- MAXF760102** Normalized frequency of extended structure (Maxfield-Scheraga, 1976)
- MAXF760103** Normalized frequency of zeta R (Maxfield-Scheraga, 1976)
- MAXF760104** Normalized frequency of left-handed alpha-helix (Maxfield-Scheraga, 1976)
- MAXF760105** Normalized frequency of zeta L (Maxfield-Scheraga, 1976)
- MAXF760106** Normalized frequency of alpha region (Maxfield-Scheraga, 1976)
- MCMT640101** Refractivity (McMeekin et al., 1964), Cited by Jones (1975)
- MEEJ800101** Retention coefficient in HPLC, pH7.4 (Meek, 1980)
- MEEJ800102** Retention coefficient in HPLC, pH2.1 (Meek, 1980)
- MEEJ810101** Retention coefficient in NaClO₄ (Meek-Rossetti, 1981)
- MEEJ810102** Retention coefficient in NaH₂PO₄ (Meek-Rossetti, 1981)
- MEIH800101** Average reduced distance for C-alpha (Meirovitch et al., 1980)
- MEIH800102** Average reduced distance for side chain (Meirovitch et al., 1980)
- MEIH800103** Average side chain orientation angle (Meirovitch et al., 1980)
- MIYS850101** Effective partition energy (Miyazawa-Jernigan, 1985)
- NAGK730101** Normalized frequency of alpha-helix (Nagano, 1973)
- NAGK730102** Normalized frequency of beta-structure (Nagano, 1973)
- NAGK730103** Normalized frequency of coil (Nagano, 1973)
- NAKH900101** AA composition of total proteins (Nakashima et al., 1990)
- NAKH900102** SD of AA composition of total proteins (Nakashima et al., 1990)
- NAKH900103** AA composition of mt-proteins (Nakashima et al., 1990)
- NAKH900104** Normalized composition of mt-proteins (Nakashima et al., 1990)
- NAKH900105** AA composition of mt-proteins from animal (Nakashima et al., 1990)
- NAKH900106** Normalized composition from animal (Nakashima et al., 1990)
- NAKH900107** AA composition of mt-proteins from fungi and plant (Nakashima et al., 1990)
- NAKH900108** Normalized composition from fungi and plant (Nakashima et al., 1990)
- NAKH900109** AA composition of membrane proteins (Nakashima et al., 1990)
- NAKH900110** Normalized composition of membrane proteins (Nakashima et al., 1990)
- NAKH900111** Transmembrane regions of non-mt-proteins (Nakashima et al., 1990)
- NAKH900112** Transmembrane regions of mt-proteins (Nakashima et al., 1990)
- NAKH900113** Ratio of average and computed composition (Nakashima et al., 1990)
- NAKH920101** AA composition of CYT of single-spanning proteins (Nakashima-Nishikawa, 1992)
- NAKH920102** AA composition of CYT2 of single-spanning proteins (Nakashima-Nishikawa, 1992)
- NAKH920103** AA composition of EXT of single-spanning proteins (Nakashima-Nishikawa, 1992)
- NAKH920104** AA composition of EXT2 of single-spanning proteins (Nakashima-Nishikawa, 1992)
- NAKH920105** AA composition of MEM of single-spanning proteins (Nakashima-Nishikawa, 1992)
- NAKH920106** AA composition of CYT of multi-spanning proteins (Nakashima-Nishikawa, 1992)

NAKH920107 AA composition of EXT of multi-spanning proteins (Nakashima-Nishikawa, 1992)
NAKH920108 AA composition of MEM of multi-spanning proteins (Nakashima-Nishikawa, 1992)
NISK800101 8 A contact number (Nishikawa-Ooi, 1980)
NISK860101 14 A contact number (Nishikawa-Ooi, 1986)
NOZY710101 Transfer energy, organic solvent/water (Nozaki-Tanford, 1971)
OOBM770101 Average non-bonded energy per atom (Oobatake-Ooi, 1977)
OOBM770102 Short and medium range non-bonded energy per atom (Oobatake-Ooi, 1977)
OOBM770103 Long range non-bonded energy per atom (Oobatake-Ooi, 1977)
OOBM770104 Average non-bonded energy per residue (Oobatake-Ooi, 1977)
OOBM770105 Short and medium range non-bonded energy per residue (Oobatake-Ooi, 1977)
OOBM850101 Optimized beta-structure-coil equilibrium constant (Oobatake et al., 1985)
OOBM850102 Optimized propensity to form reverse turn (Oobatake et al., 1985)
OOBM850103 Optimized transfer energy parameter (Oobatake et al., 1985)
OOBM850104 Optimized average non-bonded energy per atom (Oobatake et al., 1985)
OOBM850105 Optimized side chain interaction parameter (Oobatake et al., 1985)
PALJ810101 Normalized frequency of alpha-helix from LG (Palau et al., 1981)
PALJ810102 Normalized frequency of alpha-helix from CF (Palau et al., 1981)
PALJ810103 Normalized frequency of beta-sheet from LG (Palau et al., 1981)
PALJ810104 Normalized frequency of beta-sheet from CF (Palau et al., 1981)
PALJ810105 Normalized frequency of turn from LG (Palau et al., 1981)
PALJ810106 Normalized frequency of turn from CF (Palau et al., 1981)
PALJ810107 Normalized frequency of alpha-helix in all-alpha class (Palau et al., 1981)
PALJ810108 Normalized frequency of alpha-helix in alpha+beta class (Palau et al., 1981)
PALJ810109 Normalized frequency of alpha-helix in alpha/beta class (Palau et al., 1981)
PALJ810110 Normalized frequency of beta-sheet in all-beta class (Palau et al., 1981)
PALJ810111 Normalized frequency of beta-sheet in alpha+beta class (Palau et al., 1981)
PALJ810112 Normalized frequency of beta-sheet in alpha/beta class (Palau et al., 1981)
PALJ810113 Normalized frequency of turn in all-alpha class (Palau et al., 1981)
PALJ810114 Normalized frequency of turn in all-beta class (Palau et al., 1981)
PALJ810115 Normalized frequency of turn in alpha+beta class (Palau et al., 1981)
PALJ810116 Normalized frequency of turn in alpha/beta class (Palau et al., 1981)
PARJ860101 HPLC parameter (Parker et al., 1986)
PLIV810101 Partition coefficient (Pliska et al., 1981)
PONP800101 Surrounding hydrophobicity in folded form (Ponnuswamy et al., 1980)
PONP800102 Average gain in surrounding hydrophobicity (Ponnuswamy et al., 1980)
PONP800103 Average gain ratio in surrounding hydrophobicity (Ponnuswamy et al., 1980)
PONP800104 Surrounding hydrophobicity in alpha-helix (Ponnuswamy et al., 1980)

- PONP800105** Surrounding hydrophobicity in beta-sheet (Ponnuswamy et al., 1980)
- PONP800106** Surrounding hydrophobicity in turn (Ponnuswamy et al., 1980)
- PONP800107** Accessibility reduction ratio (Ponnuswamy et al., 1980)
- PONP800108** Average number of surrounding residues (Ponnuswamy et al., 1980)
- PRAM820101** Intercept in regression analysis (Prabhakaran-Ponnuswamy, 1982)
- PRAM820102** Slope in regression analysis $\times 1.0E1$ (Prabhakaran-Ponnuswamy, 1982)
- PRAM820103** Correlation coefficient in regression analysis (Prabhakaran-Ponnuswamy, 1982)
- PRAM900101** Hydrophobicity (Prabhakaran, 1990)
- PRAM900102** Relative frequency in alpha-helix (Prabhakaran, 1990)
- PRAM900103** Relative frequency in beta-sheet (Prabhakaran, 1990)
- PRAM900104** Relative frequency in reverse-turn (Prabhakaran, 1990)
- PTIO830101** Helix-coil equilibrium constant (Ptitsyn-Finkelstein, 1983)
- PTIO830102** Beta-coil equilibrium constant (Ptitsyn-Finkelstein, 1983)
- QIAN880101** Weights for alpha-helix at the window position of -6 (Qian-Sejnowski, 1988)
- QIAN880102** Weights for alpha-helix at the window position of -5 (Qian-Sejnowski, 1988)
- QIAN880103** Weights for alpha-helix at the window position of -4 (Qian-Sejnowski, 1988)
- QIAN880104** Weights for alpha-helix at the window position of -3 (Qian-Sejnowski, 1988)
- QIAN880105** Weights for alpha-helix at the window position of -2 (Qian-Sejnowski, 1988)
- QIAN880106** Weights for alpha-helix at the window position of -1 (Qian-Sejnowski, 1988)
- QIAN880107** Weights for alpha-helix at the window position of 0 (Qian-Sejnowski, 1988)
- QIAN880108** Weights for alpha-helix at the window position of 1 (Qian-Sejnowski, 1988)
- QIAN880109** Weights for alpha-helix at the window position of 2 (Qian-Sejnowski, 1988)
- QIAN880110** Weights for alpha-helix at the window position of 3 (Qian-Sejnowski, 1988)
- QIAN880111** Weights for alpha-helix at the window position of 4 (Qian-Sejnowski, 1988)
- QIAN880112** Weights for alpha-helix at the window position of 5 (Qian-Sejnowski, 1988)
- QIAN880113** Weights for alpha-helix at the window position of 6 (Qian-Sejnowski, 1988)
- QIAN880114** Weights for beta-sheet at the window position of -6 (Qian-Sejnowski, 1988)
- QIAN880115** Weights for beta-sheet at the window position of -5 (Qian-Sejnowski, 1988)
- QIAN880116** Weights for beta-sheet at the window position of -4 (Qian-Sejnowski, 1988)
- QIAN880117** Weights for beta-sheet at the window position of -3 (Qian-Sejnowski, 1988)
- QIAN880118** Weights for beta-sheet at the window position of -2 (Qian-Sejnowski, 1988)
- QIAN880119** Weights for beta-sheet at the window position of -1 (Qian-Sejnowski, 1988)
- QIAN880120** Weights for beta-sheet at the window position of 0 (Qian-Sejnowski, 1988)
- QIAN880121** Weights for beta-sheet at the window position of 1 (Qian-Sejnowski, 1988)
- QIAN880122** Weights for beta-sheet at the window position of 2 (Qian-Sejnowski, 1988)
- QIAN880123** Weights for beta-sheet at the window position of 3 (Qian-Sejnowski, 1988)
- QIAN880124** Weights for beta-sheet at the window position of 4 (Qian-Sejnowski, 1988)

- QIAN880125** Weights for beta-sheet at the window position of 5 (Qian-Sejnowski, 1988)
- QIAN880126** Weights for beta-sheet at the window position of 6 (Qian-Sejnowski, 1988)
- QIAN880127** Weights for coil at the window position of -6 (Qian-Sejnowski, 1988)
- QIAN880128** Weights for coil at the window position of -5 (Qian-Sejnowski, 1988)
- QIAN880129** Weights for coil at the window position of -4 (Qian-Sejnowski, 1988)
- QIAN880130** Weights for coil at the window position of -3 (Qian-Sejnowski, 1988)
- QIAN880131** Weights for coil at the window position of -2 (Qian-Sejnowski, 1988)
- QIAN880132** Weights for coil at the window position of -1 (Qian-Sejnowski, 1988)
- QIAN880133** Weights for coil at the window position of 0 (Qian-Sejnowski, 1988)
- QIAN880134** Weights for coil at the window position of 1 (Qian-Sejnowski, 1988)
- QIAN880135** Weights for coil at the window position of 2 (Qian-Sejnowski, 1988)
- QIAN880136** Weights for coil at the window position of 3 (Qian-Sejnowski, 1988)
- QIAN880137** Weights for coil at the window position of 4 (Qian-Sejnowski, 1988)
- QIAN880138** Weights for coil at the window position of 5 (Qian-Sejnowski, 1988)
- QIAN880139** Weights for coil at the window position of 6 (Qian-Sejnowski, 1988)
- RACS770101** Average reduced distance for C-alpha (Rackovsky-Scheraga, 1977)
- RACS770102** Average reduced distance for side chain (Rackovsky-Scheraga, 1977)
- RACS770103** Side chain orientational preference (Rackovsky-Scheraga, 1977)
- RACS820101** Average relative fractional occurrence in A0(i) (Rackovsky-Scheraga, 1982)
- RACS820102** Average relative fractional occurrence in AR(i) (Rackovsky-Scheraga, 1982)
- RACS820103** Average relative fractional occurrence in AL(i) (Rackovsky-Scheraga, 1982)
- RACS820104** Average relative fractional occurrence in EL(i) (Rackovsky-Scheraga, 1982)
- RACS820105** Average relative fractional occurrence in E0(i) (Rackovsky-Scheraga, 1982)
- RACS820106** Average relative fractional occurrence in ER(i) (Rackovsky-Scheraga, 1982)
- RACS820107** Average relative fractional occurrence in A0(i-1) (Rackovsky-Scheraga, 1982)
- RACS820108** Average relative fractional occurrence in AR(i-1) (Rackovsky-Scheraga, 1982)
- RACS820109** Average relative fractional occurrence in AL(i-1) (Rackovsky-Scheraga, 1982)
- RACS820110** Average relative fractional occurrence in EL(i-1) (Rackovsky-Scheraga, 1982)
- RACS820111** Average relative fractional occurrence in E0(i-1) (Rackovsky-Scheraga, 1982)
- RACS820112** Average relative fractional occurrence in ER(i-1) (Rackovsky-Scheraga, 1982)
- RACS820113** Value of theta(i) (Rackovsky-Scheraga, 1982)
- RACS820114** Value of theta(i-1) (Rackovsky-Scheraga, 1982)
- RADA880101** Transfer free energy from chx to wat (Radzicka-Wolfenden, 1988)
- RADA880102** Transfer free energy from oct to wat (Radzicka-Wolfenden, 1988)
- RADA880103** Transfer free energy from vap to chx (Radzicka-Wolfenden, 1988)
- RADA880104** Transfer free energy from chx to oct (Radzicka-Wolfenden, 1988)
- RADA880105** Transfer free energy from vap to oct (Radzicka-Wolfenden, 1988)

RADA880106 Accessible surface area (Radzicka-Wolfenden, 1988)
RADA880107 Energy transfer from out to in(95%buried) (Radzicka-Wolfenden, 1988)
RADA880108 Mean polarity (Radzicka-Wolfenden, 1988)
RICJ880101 Relative preference value at N" (Richardson-Richardson, 1988)
RICJ880102 Relative preference value at N' (Richardson-Richardson, 1988)
RICJ880103 Relative preference value at N-cap (Richardson-Richardson, 1988)
RICJ880104 Relative preference value at N1 (Richardson-Richardson, 1988)
RICJ880105 Relative preference value at N2 (Richardson-Richardson, 1988)
RICJ880106 Relative preference value at N3 (Richardson-Richardson, 1988)
RICJ880107 Relative preference value at N4 (Richardson-Richardson, 1988)
RICJ880108 Relative preference value at N5 (Richardson-Richardson, 1988)
RICJ880109 Relative preference value at Mid (Richardson-Richardson, 1988)
RICJ880110 Relative preference value at C5 (Richardson-Richardson, 1988)
RICJ880111 Relative preference value at C4 (Richardson-Richardson, 1988)
RICJ880112 Relative preference value at C3 (Richardson-Richardson, 1988)
RICJ880113 Relative preference value at C2 (Richardson-Richardson, 1988)
RICJ880114 Relative preference value at C1 (Richardson-Richardson, 1988)
RICJ880115 Relative preference value at C-cap (Richardson-Richardson, 1988)
RICJ880116 Relative preference value at C' (Richardson-Richardson, 1988)
RICJ880117 Relative preference value at C" (Richardson-Richardson, 1988)
ROBB760101 Information measure for alpha-helix (Robson-Suzuki, 1976)
ROBB760102 Information measure for N-terminal helix (Robson-Suzuki, 1976)
ROBB760103 Information measure for middle helix (Robson-Suzuki, 1976)
ROBB760104 Information measure for C-terminal helix (Robson-Suzuki, 1976)
ROBB760105 Information measure for extended (Robson-Suzuki, 1976)
ROBB760106 Information measure for pleated-sheet (Robson-Suzuki, 1976)
ROBB760107 Information measure for extended without H-bond (Robson-Suzuki, 1976)
ROBB760108 Information measure for turn (Robson-Suzuki, 1976)
ROBB760109 Information measure for N-terminal turn (Robson-Suzuki, 1976)
ROBB760110 Information measure for middle turn (Robson-Suzuki, 1976)
ROBB760111 Information measure for C-terminal turn (Robson-Suzuki, 1976)
ROBB760112 Information measure for coil (Robson-Suzuki, 1976)
ROBB760113 Information measure for loop (Robson-Suzuki, 1976)
ROBB790101 Hydration free energy (Robson-Osguthorpe, 1979)
ROSG850101 Mean area buried on transfer (Rose et al., 1985)
ROSG850102 Mean fractional area loss (Rose et al., 1985)
ROSM880101 Side chain hydrophathy, uncorrected for solvation (Roseman, 1988)

ROSM880102 Side chain hydrophathy, corrected for solvation (Roseman, 1988)
ROSM880103 Loss of Side chain hydrophathy by helix formation (Roseman, 1988)
SIMZ760101 Transfer free energy (Simon, 1976), Cited by Charton-Charton (1982)
SNEP660101 Principal component I (Sneath, 1966)
SNEP660102 Principal component II (Sneath, 1966)
SNEP660103 Principal component III (Sneath, 1966)
SNEP660104 Principal component IV (Sneath, 1966)
SUEM840101 Zimm-Bragg parameter s at 20 C (Sueki et al., 1984)
SUEM840102 Zimm-Bragg parameter $\sigma \times 1.0E4$ (Sueki et al., 1984)
SWER830101 Optimal matching hydrophobicity (Sweet-Eisenberg, 1983)
TANS770101 Normalized frequency of alpha-helix (Tanaka-Scheraga, 1977)
TANS770102 Normalized frequency of isolated helix (Tanaka-Scheraga, 1977)
TANS770103 Normalized frequency of extended structure (Tanaka-Scheraga, 1977)
TANS770104 Normalized frequency of chain reversal R (Tanaka-Scheraga, 1977)
TANS770105 Normalized frequency of chain reversal S (Tanaka-Scheraga, 1977)
TANS770106 Normalized frequency of chain reversal D (Tanaka-Scheraga, 1977)
TANS770107 Normalized frequency of left-handed helix (Tanaka-Scheraga, 1977)
TANS770108 Normalized frequency of zeta R (Tanaka-Scheraga, 1977)
TANS770109 Normalized frequency of coil (Tanaka-Scheraga, 1977)
TANS770110 Normalized frequency of chain reversal (Tanaka-Scheraga, 1977)
VASM830101 Relative population of conformational state A (Vasquez et al., 1983)
VASM830102 Relative population of conformational state C (Vasquez et al., 1983)
VASM830103 Relative population of conformational state E (Vasquez et al., 1983)
VELV850101 Electron-ion interaction potential (Veljkovic et al., 1985)
VENT840101 Bitterness (Venanzi, 1984)
VHEG790101 Transfer free energy to lipophilic phase (von Heijne-Blomberg, 1979)
WARP780101 Average interactions per side chain atom (Warne-Morgan, 1978)
WEBA780101 RF value in high salt chromatography (Weber-Lacey, 1978)
WERD780101 Propensity to be buried inside (Wertz-Scheraga, 1978)
WERD780102 Free energy change of $\epsilon(i)$ to $\epsilon(ex)$ (Wertz-Scheraga, 1978)
WERD780103 Free energy change of $\alpha(Ri)$ to $\alpha(Rh)$ (Wertz-Scheraga, 1978)
WERD780104 Free energy change of $\epsilon(i)$ to $\alpha(Rh)$ (Wertz-Scheraga, 1978)
WOEC730101 Polar requirement (Woese, 1973)
WOLR810101 Hydration potential (Wolfenden et al., 1981)
WOLS870101 Principal property value z_1 (Wold et al., 1987)
WOLS870102 Principal property value z_2 (Wold et al., 1987)
WOLS870103 Principal property value z_3 (Wold et al., 1987)

- YUTK870101** Unfolding Gibbs energy in water, pH7.0 (Yutani et al., 1987)
- YUTK870102** Unfolding Gibbs energy in water, pH9.0 (Yutani et al., 1987)
- YUTK870103** Activation Gibbs energy of unfolding, pH7.0 (Yutani et al., 1987)
- YUTK870104** Activation Gibbs energy of unfolding, pH9.0 (Yutani et al., 1987)
- ZASB820101** Dependence of partition coefficient on ionic strength (Zaslavsky et al., 1982)
- ZIMJ680101** Hydrophobicity (Zimmerman et al., 1968)
- ZIMJ680102** Bulkiness (Zimmerman et al., 1968)
- ZIMJ680103** Polarity (Zimmerman et al., 1968)
- ZIMJ680104** Isoelectric point (Zimmerman et al., 1968)
- ZIMJ680105** RF rank (Zimmerman et al., 1968)
- AURR980101** Normalized positional residue frequency at helix termini N4' (Aurora-Rose, 1998)
- AURR980102** Normalized positional residue frequency at helix termini N''' (Aurora-Rose, 1998)
- AURR980103** Normalized positional residue frequency at helix termini N'' (Aurora-Rose, 1998)
- AURR980104** Normalized positional residue frequency at helix termini N' (Aurora-Rose, 1998)
- AURR980105** Normalized positional residue frequency at helix termini Nc (Aurora-Rose, 1998)
- AURR980106** Normalized positional residue frequency at helix termini N1 (Aurora-Rose, 1998)
- AURR980107** Normalized positional residue frequency at helix termini N2 (Aurora-Rose, 1998)
- AURR980108** Normalized positional residue frequency at helix termini N3 (Aurora-Rose, 1998)
- AURR980109** Normalized positional residue frequency at helix termini N4 (Aurora-Rose, 1998)
- AURR980110** Normalized positional residue frequency at helix termini N5 (Aurora-Rose, 1998)
- AURR980111** Normalized positional residue frequency at helix termini C5 (Aurora-Rose, 1998)
- AURR980112** Normalized positional residue frequency at helix termini C4 (Aurora-Rose, 1998)
- AURR980113** Normalized positional residue frequency at helix termini C3 (Aurora-Rose, 1998)
- AURR980114** Normalized positional residue frequency at helix termini C2 (Aurora-Rose, 1998)
- AURR980115** Normalized positional residue frequency at helix termini C1 (Aurora-Rose, 1998)
- AURR980116** Normalized positional residue frequency at helix termini Cc (Aurora-Rose, 1998)
- AURR980117** Normalized positional residue frequency at helix termini C' (Aurora-Rose, 1998)
- AURR980118** Normalized positional residue frequency at helix termini C'' (Aurora-Rose, 1998)
- AURR980119** Normalized positional residue frequency at helix termini C''' (Aurora-Rose, 1998)
- AURR980120** Normalized positional residue frequency at helix termini C4' (Aurora-Rose, 1998)
- ONEK900101** Delta G values for the peptides extrapolated to 0 M urea (O'Neil-DeGrado, 1990)
- ONEK900102** Helix formation parameters (delta delta G) (O'Neil-DeGrado, 1990)
- VINM940101** Normalized flexibility parameters (B-values), average (Vihinen et al., 1994)
- VINM940102** Normalized flexibility parameters (B-values) for each residue surrounded by none rigid neighbours (Vihinen et al., 1994)
- VINM940103** Normalized flexibility parameters (B-values) for each residue surrounded by one rigid neighbours (Vihinen et al., 1994)

- VINM940104** Normalized flexibility parameters (B-values) for each residue surrounded by two rigid neighbours (Vihinen et al., 1994)
- MUNV940101** Free energy in alpha-helical conformation (Munoz-Serrano, 1994)
- MUNV940102** Free energy in alpha-helical region (Munoz-Serrano, 1994)
- MUNV940103** Free energy in beta-strand conformation (Munoz-Serrano, 1994)
- MUNV940104** Free energy in beta-strand region (Munoz-Serrano, 1994)
- MUNV940105** Free energy in beta-strand region (Munoz-Serrano, 1994)
- WIMW960101** Free energies of transfer of AcWl-X-LL peptides from bilayer interface to water (Wimley-White, 1996)
- KIMC930101** Thermodynamic beta sheet propensity (Kim-Berg, 1993)
- MONM990101** Turn propensity scale for transmembrane helices (Monne et al., 1999)
- BLAM930101** Alpha helix propensity of position 44 in T4 lysozyme (Blaber et al., 1993)
- PARS000101** p-Values of mesophilic proteins based on the distributions of B values (Parthasarathy-Murthy, 2000)
- PARS000102** p-Values of thermophilic proteins based on the distributions of B values (Parthasarathy-Murthy, 2000)
- KUMS000101** Distribution of amino acid residues in the 18 non-redundant families of thermophilic proteins (Kumar et al., 2000)
- KUMS000102** Distribution of amino acid residues in the 18 non-redundant families of mesophilic proteins (Kumar et al., 2000)
- KUMS000103** Distribution of amino acid residues in the alpha-helices in thermophilic proteins (Kumar et al., 2000)
- KUMS000104** Distribution of amino acid residues in the alpha-helices in mesophilic proteins (Kumar et al., 2000)
- TAKK010101** Side-chain contribution to protein stability (kJ/mol) (Takano-Yutani, 2001)
- FODM020101** Propensity of amino acids within pi-helices (Fodje-Al-Karadaghi, 2002)
- NADH010101** Hydrophathy scale based on self-information values in the two-state model (5% accessibility) (Naderi-Manesh et al., 2001)
- NADH010102** Hydrophathy scale based on self-information values in the two-state model (9% accessibility) (Naderi-Manesh et al., 2001)
- NADH010103** Hydrophathy scale based on self-information values in the two-state model (16% accessibility) (Naderi-Manesh et al., 2001)
- NADH010104** Hydrophathy scale based on self-information values in the two-state model (20% accessibility) (Naderi-Manesh et al., 2001)
- NADH010105** Hydrophathy scale based on self-information values in the two-state model (25% accessibility) (Naderi-Manesh et al., 2001)
- NADH010106** Hydrophathy scale based on self-information values in the two-state model (36% accessibility) (Naderi-Manesh et al., 2001)
- NADH010107** Hydrophathy scale based on self-information values in the two-state model (50% accessibility) (Naderi-Manesh et al., 2001)
- MONM990201** Averaged turn propensities in a transmembrane helix (Monne et al., 1999)

- KOEP990101** Alpha-helix propensity derived from designed sequences (Koehl-Levitt, 1999)
- KOEP990102** Beta-sheet propensity derived from designed sequences (Koehl-Levitt, 1999)
- CEDJ970101** Composition of amino acids in extracellular proteins (percent) (Cedano et al., 1997)
- CEDJ970102** Composition of amino acids in anchored proteins (percent) (Cedano et al., 1997)
- CEDJ970103** Composition of amino acids in membrane proteins (percent) (Cedano et al., 1997)
- CEDJ970104** Composition of amino acids in intracellular proteins (percent) (Cedano et al., 1997)
- CEDJ970105** Composition of amino acids in nuclear proteins (percent) (Cedano et al., 1997)
- FUKS010101** Surface composition of amino acids in intracellular proteins of thermophiles (percent) (Fukuchi-Nishikawa, 2001)
- FUKS010102** Surface composition of amino acids in intracellular proteins of mesophiles (percent) (Fukuchi-Nishikawa, 2001)
- FUKS010103** Surface composition of amino acids in extracellular proteins of mesophiles (percent) (Fukuchi-Nishikawa, 2001)
- FUKS010104** Surface composition of amino acids in nuclear proteins (percent) (Fukuchi-Nishikawa, 2001)
- FUKS010105** Interior composition of amino acids in intracellular proteins of thermophiles (percent) (Fukuchi-Nishikawa, 2001)
- FUKS010106** Interior composition of amino acids in intracellular proteins of mesophiles (percent) (Fukuchi-Nishikawa, 2001)
- FUKS010107** Interior composition of amino acids in extracellular proteins of mesophiles (percent) (Fukuchi-Nishikawa, 2001)
- FUKS010108** Interior composition of amino acids in nuclear proteins (percent) (Fukuchi-Nishikawa, 2001)
- FUKS010109** Entire chain composition of amino acids in intracellular proteins of thermophiles (percent) (Fukuchi-Nishikawa, 2001)
- FUKS010110** Entire chain composition of amino acids in intracellular proteins of mesophiles (percent) (Fukuchi-Nishikawa, 2001)
- FUKS010111** Entire chain composition of amino acids in extracellular proteins of mesophiles (percent) (Fukuchi-Nishikawa, 2001)
- FUKS010112** Entire chain composition of amino acids in nuclear proteins (percent) (Fukuchi-Nishikawa, 2001)
- AVBF000101** Screening coefficients gamma, local (Avbelj, 2000)
- AVBF000102** Screening coefficients gamma, non-local (Avbelj, 2000)
- AVBF000103** Slopes tripeptide, FDPB VFF neutral (Avbelj, 2000)
- AVBF000104** Slopes tripeptides, LD VFF neutral (Avbelj, 2000)
- AVBF000105** Slopes tripeptide, FDPB VFF noside (Avbelj, 2000)
- AVBF000106** Slopes tripeptide FDPB VFF all (Avbelj, 2000)
- AVBF000107** Slopes tripeptide FDPB PARSE neutral (Avbelj, 2000)
- AVBF000108** Slopes decapeptide, FDPB VFF neutral (Avbelj, 2000)
- AVBF000109** Slopes proteins, FDPB VFF neutral (Avbelj, 2000)

- YANJ020101** Side-chain conformation by gaussian evolutionary method (Yang et al., 2002)
- MITSO20101** Amphiphilicity index (Mitaku et al., 2002)
- TSAJ990101** Volumes including the crystallographic waters using the ProtOr (Tsai et al., 1999)
- TSAJ990102** Volumes not including the crystallographic waters using the ProtOr (Tsai et al., 1999)
- COSI940101** Electron-ion interaction potential values (Cotic, 1994)
- PONP930101** Hydrophobicity scales (Ponnuswamy, 1993)
- WILM950101** Hydrophobicity coefficient in RP-HPLC, C18 with 0.1%TFA/MeCN/H₂O (Wilce et al. 1995)
- WILM950102** Hydrophobicity coefficient in RP-HPLC, C8 with 0.1%TFA/MeCN/H₂O (Wilce et al. 1995)
- WILM950103** Hydrophobicity coefficient in RP-HPLC, C4 with 0.1%TFA/MeCN/H₂O (Wilce et al. 1995)
- WILM950104** Hydrophobicity coefficient in RP-HPLC, C18 with 0.1%TFA/2-PrOH/MeCN/H₂O (Wilce et al. 1995)
- KUHL950101** Hydrophilicity scale (Kuhn et al., 1995)
- GUOD860101** Retention coefficient at pH 2 (Guo et al., 1986)
- JURD980101** Modified Kyte-Doolittle hydrophobicity scale (Juretic et al., 1998)
- BASU050101** Interactivity scale obtained from the contact matrix (Bastolla et al., 2005)
- BASU050102** Interactivity scale obtained by maximizing the mean of correlation coefficient over single-domain globular proteins (Bastolla et al., 2005)
- BASU050103** Interactivity scale obtained by maximizing the mean of correlation coefficient over pairs of sequences sharing the TIM barrel fold (Bastolla et al., 2005)
- SUYM030101** Linker propensity index (Suyama-Ohara, 2003)
- PUNT030101** Knowledge-based membrane-propensity scale from 1D_Helix in MPtopo databases (Punta-Maritan, 2003)
- PUNT030102** Knowledge-based membrane-propensity scale from 3D_Helix in MPtopo databases (Punta-Maritan, 2003)
- GEOR030101** Linker propensity from all dataset (George-Heringa, 2003)
- GEOR030102** Linker propensity from 1-linker dataset (George-Heringa, 2003)
- GEOR030103** Linker propensity from 2-linker dataset (George-Heringa, 2003)
- GEOR030104** Linker propensity from 3-linker dataset (George-Heringa, 2003)
- GEOR030105** Linker propensity from small dataset (linker length is less than six residues) (George-Heringa, 2003)
- GEOR030106** Linker propensity from medium dataset (linker length is between six and 14 residues) (George-Heringa, 2003)
- GEOR030107** Linker propensity from long dataset (linker length is greater than 14 residues) (George-Heringa, 2003)
- GEOR030108** Linker propensity from helical (annotated by DSSP) dataset (George-Heringa, 2003)
- GEOR030109** Linker propensity from non-helical (annotated by DSSP) dataset (George-Heringa, 2003)

- ZHOH040101** The stability scale from the knowledge-based atom-atom potential (Zhou-Zhou, 2004)
- ZHOH040102** The relative stability scale extracted from mutation experiments (Zhou-Zhou, 2004)
- ZHOH040103** Buriability (Zhou-Zhou, 2004)
- BAEK050101** Linker index (Bae et al., 2005)
- HARY940101** Mean volumes of residues buried in protein interiors (Harpaz et al., 1994)
- PONJ960101** Average volumes of residues (Pontius et al., 1996)
- DIGM050101** Hydrostatic pressure asymmetry index, PAI (Di Giulio, 2005)
- WOLR790101** Hydrophobicity index (Wolfenden et al., 1979)
- OLSK800101** Average internal preferences (Olsen, 1980)
- KIDA850101** Hydrophobicity-related index (Kidera et al., 1985)
- GUYH850102** Apparent partition energies calculated from Wertz-Scheraga index (Guy, 1985)
- GUYH850103** Apparent partition energies calculated from Robson-Osguthorpe index (Guy, 1985)
- GUYH850104** Apparent partition energies calculated from Janin index (Guy, 1985)
- GUYH850105** Apparent partition energies calculated from Chothia index (Guy, 1985)
- ROSM880104** Hydropathies of amino acid side chains, neutral form (Roseman, 1988)
- ROSM880105** Hydropathies of amino acid side chains, pi-values in pH 7.0 (Roseman, 1988)
- JACR890101** Weights from the IFH scale (Jacobs-White, 1989)
- COWR900101** Hydrophobicity index, 3.0 pH (Cowan-Whittaker, 1990)
- BLAS910101** Scaled side chain hydrophobicity values (Black-Mould, 1991)
- CASG920101** Hydrophobicity scale from native protein structures (Casari-Sippl, 1992)
- CORJ870101** NNEIG index (Cornette et al., 1987)
- CORJ870102** SWEIG index (Cornette et al., 1987)
- CORJ870103** PRIFT index (Cornette et al., 1987)
- CORJ870104** PRILS index (Cornette et al., 1987)
- CORJ870105** ALTFT index (Cornette et al., 1987)
- CORJ870106** ALTLS index (Cornette et al., 1987)
- CORJ870107** TOTFT index (Cornette et al., 1987)
- CORJ870108** TOTLS index (Cornette et al., 1987)
- MIYS990101** Relative partition energies derived by the Bethe approximation (Miyazawa-Jernigan, 1999)
- MIYS990102** Optimized relative partition energies - method A (Miyazawa-Jernigan, 1999)
- MIYS990103** Optimized relative partition energies - method B (Miyazawa-Jernigan, 1999)
- MIYS990104** Optimized relative partition energies - method C (Miyazawa-Jernigan, 1999)
- MIYS990105** Optimized relative partition energies - method D (Miyazawa-Jernigan, 1999)
- ENGD860101** Hydrophobicity index (Engelman et al., 1986)
- FASG890101** Hydrophobicity index (Fasman, 1989)
- K6.5** Values of W_c in proteins from class Beta, cutoff 6 A, separation 5 (Wozniak, 2014)

- K8.5** Values of Wc in proteins from class Beta, cutoff 8 A, separation 5 (Wozniak, 2014)
- K12.5** Values of Wc in proteins from class Beta, cutoff 12 A, separation 5 (Wozniak, 2014)
- K6.15** Values of Wc in proteins from class Beta, cutoff 6 A, separation 15 (Wozniak, 2014)
- K8.15** Values of Wc in proteins from class Beta, cutoff 8 A, separation 15 (Wozniak, 2014)
- K12.15** Values of Wc in proteins from class Beta, cutoff 12 A, separation 15 (Wozniak, 2014)

Source

AAIndex database.

References

- Kawashima, S. and Kanehisa, M. (2000) AAindex: amino acid index database. *Nucleic Acids Res.*, 28:374.
- Wozniak, P. and Kotulska M. (2014) Characteristics of protein residue-residue contacts and their application in contact prediction. 20(11):2497

Examples

```
data(aaprop)
```

add_1grams	<i>Add 1-grams</i>
------------	--------------------

Description

Builds (n+1)-grams from n-grams.

Usage

```
add_1grams(ngram, u, seq_length)
```

Arguments

ngram	a single n-gram.
u	integer, numeric or character vector of all possible unigrams.
seq_length	length of an origin sequence.

Details

n-grams are built by pasting every possible unigram in the every possible free position. The total length of n-gram (n plus total distance between elements of the n-gram) is limited by the length of an origin sequence, because the n-gram cannot be longer than an origin sequence.

Value

vector of n-grams (where n is equal to the n of the input plus one).

See Also

Reverse function: [gap_ngrams](#).

Examples

```
add_1grams("1_2.3.4_3.0", 1L:4, 8)
```

```
add_1grams("a.a_1", c("a", "b", "c"), 4)
```

```
as.data.frame.feature_test
```

Coerce feature_test object to a data frame

Description

Coerce results of [test_features](#) function to a [data.frame](#).

Usage

```
## S3 method for class 'feature_test'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  stringsAsFactors = FALSE,
  ...
)
```

Arguments

x	object of class feature_test .
row.names	ignored.
optional	ignored.
stringsAsFactors	logical: should the character vector be converted to a factor?.
...	additional arguments to be passed to or from methods.

Value

a data frame with four columns: names of n-gram, p-values, occurrences in positive and negative sequences.

binarize	<i>Binarize</i>
----------	-----------------

Description

Binarizes a matrix.

Usage

```
binarize(x)
```

Arguments

x matrix or [simple_triplet_matrix](#).

Value

a matrix or [simple_triplet_matrix](#) (depending on the input).

calc_criterion	<i>Calculate value of criterion</i>
----------------	-------------------------------------

Description

Computes a chosen statistical criterion for each feature versus target vector.

Usage

```
calc_criterion(target, features, criterion_function)
```

Arguments

target integer vector with target information (e.g. class labels).
 features integer matrix of features with number of rows equal to the length of the target vector.
 criterion_function a function calculating criterion. For a full list, see [test_features](#).

Details

The permutation test implemented in `biogram` uses several criteria to filter important features. Each can be used by [test_features](#) by specifying the `criterion` parameter.

Value

a integer vector of length equal to the number of features containing computed information gain values.

Note

Both target and features must be binary, i.e. contain only 0 and 1 values.

See Also

[test_features](#).

Examples

```
tar <- sample(0L:1, 100, replace = TRUE)
feats <- matrix(sample(0L:1, 400, replace = TRUE), ncol = 4)

# Information Gain
calc_criterion(tar, feats, calc_ig)

# hi-squared-based measure
calc_criterion(tar, feats, calc_cs)

# Kullback-Leibler divergence
calc_criterion(tar, feats, calc_kl)
```

calc_cs

Calculate Chi-squared-based measure

Description

Computes Chi-squared-based measure between features and target vector.

Usage

```
calc_cs(feature, target, len_target, pos_target)
```

Arguments

feature	feature vector.
target	target.
len_target	length of the target vector.
pos_target	number of positive cases in the target vector.

Value

A numeric vector of length 1 representing computed Chi-square values.

Note

Both target and features must be binary, i.e. contain only 0 and 1 values.

The function was designed to be as fast as possible subroutine of [calc_criterion](#) and might be cumbersome if directly called by a user.

See Also

[test_features](#).

[chisq.test](#) - Pearson's chi-squared test for count data.

Examples

```
tar <- sample(0L:1, 100, replace = TRUE)
feat <- sample(0L:1, 100, replace = TRUE)
calc_cs(feat, tar, 100, sum(tar))
```

calc_ed

Calculate encoding distance

Description

Computes the encoding distance between two encodings.

Usage

```
calc_ed(a, b, prop = NULL, measure)
```

Arguments

- | | |
|---------|--|
| a | encoding (see validate_encoding for more information about the required structure of encoding). |
| b | encoding to which a should be compared. Must have equal number of groups or less than a. Both a and b must have the the same number of elements. |
| prop | matrix of physicochemical properties to normalize the encoding distance. Each column should represent properties of the single amino acid/nucleotide. If NULL, encoding distance is not normalized. |
| measure | character vector of length one specifying the measure. Currently available measures are "pi" (partition index) and "si" (similarity index). If the parameter prop is supplied, the encoding distance is normalized by the factor equal to the sum of distances for each group in a and the closest group in b. The position of a group is defined as the mean value of properties of amino acids or nucleotides belonging the group.
See the package vignette for more details. |

Value

an encoding distance.

See Also

[calc_si](#): compute the similarity index of two encodings. [encoding2df](#): converts an encoding to a data frame. [validate_encoding](#): validate a structure of an encoding.

Examples

```
# calculate encoding distance between two encodings of amino acids
aa1 = list(`1` = c("g", "a", "p", "v", "m", "l", "i"),
          `2` = c("k", "h"),
          `3` = c("d", "e"),
          `4` = c("f", "r", "w", "y", "s", "t", "c", "n", "q"))

aa2 = list(`1` = c("g", "a", "p", "v", "m", "l", "q"),
          `2` = c("k", "h", "d", "e", "i"),
          `3` = c("f", "r", "w", "y", "s", "t", "c", "n"))
calc_ed(aa1, aa2, measure = "pi")

# the encoding distance between two identical encodings is 0
calc_ed(aa1, aa1, measure = "pi")
```

 calc_ig

Calculate IG for single feature

Description

Computes information gain of single feature and target vector.

Usage

```
calc_ig(feature, target, len_target, pos_target)
```

Arguments

feature	feature vector.
target	target.
len_target	length of the target vector.
pos_target	number of positive cases in the target vector.

Details

The information gain term is used here (improperly) as a synonym of mutual information. It is defined as:

$$IG(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right)$$

In biogram package information gain is computed using following relationship: $IG = E(S) - E(S|F)$

Value

A numeric vector of length 1 representing information gain in nats.

Note

During calculations $0 \log 0 = 0$. For a justification see References.

The function was designed to be a fast subroutine of `calc_criterion` and might be cumbersome if directly called by a user.

References

Cover TM, Thomas JA *Elements of Information Theory, 2nd Edition* Wiley, 2006.

Examples

```
tar <- sample(0L:1, 100, replace = TRUE)
feat <- sample(0L:1, 100, replace = TRUE)
calc_ig(feat, tar, 100, sum(tar))
```

calc_kl	<i>Calculate KL divergence of features</i>
---------	--

Description

Computes Kullback-Leibler divergence between features and target vector.

Usage

```
calc_kl(feature, target, len_target, pos_target)
```

Arguments

feature	feature vector.
target	target.
len_target	length of the target vector.
pos_target	number of positive cases in the target vector.

Value

A numeric vector of length 1 representing Kullback-Leibler divergence value.

Note

Both target and features must be binary, i.e. contain only 0 and 1 values.

The function was designed to be as fast as possible subroutine of `calc_criterion` and might be cumbersome if directly called by a user.

References

Kullback S, Leibler RA *On information and sufficiency*. *Annals of Mathematical Statistics* 22 (1):79-86, 1951.

See Also

[test_features](#). Kullback-Leibler divergence is calculated using [KL.plugin](#).

Examples

```
tar <- sample(0L:1, 100, replace = TRUE)
feat <- sample(0L:1, 100, replace = TRUE)
calc_kl(feat, tar, 100, sum(tar))
```

`calc_pi`*Calculate partition index*

Description

Computes the encoding distance between two encodings.

Usage

```
calc_pi(a, b)
```

Arguments

- | | |
|---|--|
| a | encoding (see validate_encoding for more information about the required structure of encoding). |
| b | encoding to which a should be compared. Must have equal number of groups or less than a. Both a and b must have the the same number of elements. |

Details

The encoding distance between a and b is defined as the minimum number of amino acids that have to be moved between subgroups of encoding to make a identical to b (order of subgroups in the encoding and amino acids in a group is unimportant).

If the parameter `prop` is supplied, the encoding distance is normalized by the factor equal to the sum of distances for each group in a and the closest group in b. The position of a group is defined as the mean value of properties of amino acids or nucleotides belonging the group.

See the package vignette for more details.

Value

an encoding distance.

See Also

[calc_si](#): compute the similarity index of two encodings. [encoding2df](#): converts an encoding to a data frame. [validate_encoding](#): validate a structure of an encoding.

Examples

```
# calculate encoding distance between two encodings of amino acids
aa1 = list(`1` = c("g", "a", "p", "v", "m", "l", "i"),
          `2` = c("k", "h"),
          `3` = c("d", "e"),
          `4` = c("f", "r", "w", "y", "s", "t", "c", "n", "q"))

aa2 = list(`1` = c("g", "a", "p", "v", "m", "l", "q"),
          `2` = c("k", "h", "d", "e", "i"),
          `3` = c("f", "r", "w", "y", "s", "t", "c", "n"))
calc_pi(aa1, aa2)

# the encoding distance between two identical encodings is 0
calc_pi(aa1, aa1)
```

calc_si

Compute similarity index

Description

Computes similarity index between two encodings.

Usage

```
calc_si(a, b)
```

Arguments

- a encoding (see [validate_encoding](#) for more information about the required structure of encoding).
- b encoding to which a should be compared. Must have equal number of groups or less than a. Both a and b must have the the same number of elements.

Details

Briefly, the similarity index is a fraction of elements that have the same pairing in both encodings. Pairing is a binary variable, that has value 1 if two elements are in the same group and 0 if not. For more details, see references.

Value

the value of similarity index.

References

Stephenson, J.D., and Freeland, S.J. (2013). Unearthing the Root of Amino Acid Similarity. *J Mol Evol* 77, 159-169.

See Also

[calc_ed](#): calculate the encoding distance between two encodings.

Examples

```
# example from Stephenson & Freeland, 2013 (Fig. 6)
enc1 <- list(`1` = "A",
            `2` = c("F", "E"),
            `3` = c("C", "D", "G"))

enc2 <- list(`1` = c("A", "G"),
            `2` = c("C", "D", "E", "F"))

enc3 <- list(`1` = c("D", "G"),
            `2` = c("E", "F"),
            `3` = c("A", "C"))

calc_si(enc1, enc2)
calc_si(enc2, enc3)
calc_si(enc1, enc3)
```

check_criterion	<i>Check chosen criterion</i>
-----------------	-------------------------------

Description

Checks if the criterion is viable or matches it to the list of implemented criterions.

Usage

```
check_criterion(input_criterion, criterion_names = c("ig", "kl", "cs"))
```

Arguments

`input_criterion`
character string, criterion from input.

`criterion_names`
list of implemented criterions, always in lowercase.

Value

a list of three:

- criterion name,
- its function,
- nice name for outputs.

See Also

Calculate the value of criterion: [calc_criterion](#).

cluster_reg_exp	<i>Clustering of sequences based on regular expression</i>
-----------------	--

Description

Clusters sequences hierarchically with regular expressions. At each step we minimize number of degrees of freedom for all regular expressions needed to describe the data

Usage

```
cluster_reg_exp(ngrams)
```

Arguments

ngrams list of elements

Details

Regular expression is a list of the length equal to the length of the input sequences. Each element of the list represents a position in the sequence and contains amino acid, that are likely to occur on this position.

Value

List of four

- "regExps"regular expression in best clustering
- "seqClustering"clustering of sequences in best clustering
- "allRegExps"all regular expressions.
- "allIndices"all clusterings

Examples

```
data(human_cleave)
#cluster_reg_exp is computationally expensive

results <- cluster_reg_exp(human_cleave[1L:10, 1L:4])
```

code_ngrams

Code n-grams

Description

Code human-friendly representation of n-grams into a biogram format.

Usage

```
code_ngrams(decoded_ngrams)
```

Arguments

decoded_ngrams a character vector of decoded n-grams.

Value

a character vector of n-grams.

See Also

Inverse function: [decode_ngrams](#).

Examples

```
code_ngrams(c("11_2", "1__12", "222"))
code_ngrams(c("aaa_b", "d_aa", "abd"))
```

construct_ngrams*Construct and filter n-grams*

Description

Builds and selects important n-grams stepwise.

Usage

```
construct_ngrams(  
  target,  
  seq,  
  u,  
  n_max,  
  conf_level = 0.95,  
  gap = TRUE,  
  use_heuristics = TRUE  
)
```


Arguments

target	integer vector with target information (e.g. class labels).
seq	a vector or matrix describing sequence(s).
u	integer, numeric or character vector of all possible unigrams.
n_max	size of constructed n-grams.
conf_level	confidence level.
gap	logical, if TRUE gaps are used. See Details.
use_heuristics,	if FALSE then all n-grams are tested. This may slow down computations significantly

Details

construct_ngrams starts by extracting unigrams from the sequences, pasting them together in all combination and choosing from them significant features (with p-value below conf_level). The chosen n-grams are further extended to the specified by n_max size by pasting unigrams at both ends.

The gap parameter determines if construct_ngrams performs the feature selection on exact n-grams (gap equal to FALSE) or on all features in the Hamming distance 1 from the n-gram (gap equal to TRUE).

Value

a vector of n-grams.

See Also

Feature filtering method: [test_features](#).

Examples

```
# to make the example faster, we run construct_ngrams() on the
# subset of data
deg_seqs <- degenerate(human_cleave[c(1L:100, 801L:900)], 1L:9],
list(`1` = c(1, 6, 8, 10, 11, 18),
     `2` = c(2, 13, 14, 16, 17),
     `3` = c(5, 19, 20),
     `4` = c(7, 9, 12, 15),
     `5` = c(3, 4))
bigrams <- construct_ngrams(human_cleave[c(1L:100, 801L:900)], "tar", deg_seqs, 1L:5, 2)
```

count_multigrams *Detect and count multiple n-grams in sequences*

Description

A convenient wrapper around [count_ngrams](#) for counting multiple values of n and d.

Usage

```
count_multigrams(
  ns,
  ds = rep(0, length(ns)),
  seq,
  u,
  pos = FALSE,
  scale = FALSE,
  threshold = 0
)
```

Arguments

ns	numeric vector of n-grams' sizes. See Details.
ds	list of distances between elements of n-grams. Each element of the list is a vector used as distance for the respective n-gram size given by the ns parameter.
seq	a vector or matrix describing sequence(s).
u	integer, numeric or character vector of all possible unigrams.
pos	logical, if TRUE position-specific n_grams are counted.
scale	logical, if TRUE output data is normalized. May be applied only to the counts of n-grams without position information. See Details.
threshold	integer, if not equal to 0, data is binarized into two groups (larger or equal to threshold vs. smaller than threshold).

Details

ns vector and ds vector must have equal length. Elements of ds vector are used as equivalents of d parameter for respective values of ns. For example, if ns is c(4, 4, 4), the ds must be a list of length 3. Each element of the ds list must have length 3 or 1, as appropriate for a d parameter in count_ngrams function.

Value

An integer matrix with named columns. The naming conventions are the same as in [count_ngrams](#).

Examples

```
seqs <- matrix(sample(1L:4, 600, replace = TRUE), ncol = 50)
count_multigrams(c(3, 1), list(c(1, 0), 0), seqs, 1L:4, pos = TRUE)
# if ds parameter is not present, n-grams are calculated for distance 0
count_multigrams(c(3, 1), seq = seqs, u = 1L:4)

# calculate three times n-gram with the same length, but different distances between
# elements
count_multigrams(c(4, 4, 4), list(c(2, 0, 1), c(2, 1, 0), c(0, 1, 2)),
  seqs, 1L:4, pos = TRUE)
```

count_ngrams	<i>Count n-grams in sequences</i>
--------------	-----------------------------------

Description

Counts all n-grams or position-specific n-grams present in the input sequence(s).

Usage

```
count_ngrams(seq, n, u, d = 0, pos = FALSE, scale = FALSE, threshold = 0)
```

Arguments

seq	a vector or matrix describing sequence(s).
n	integer size of n-gram.
u	integer, numeric or character vector of all possible unigrams.
d	integer vector of distances between elements of n-gram (0 means consecutive elements). See Details.
pos	logical, if TRUE position-specific n_grams are counted.
scale	logical, if TRUE output data is normalized. May be applied only to the counts of n-grams without position information. See Details.
threshold	integer, if not equal to 0, data is binarized into two groups (larger or equal to threshold vs. smaller than threshold).

Details

A distance vector should be always $n - 1$ in length. For example when $n = 3$, $d = c(1,2)$ means A_A__A. For $n = 4$, $d = c(2,0,1)$ means A__AA_A. If vector d has length 1, it is recycled to length $n - 1$.

n-gram names follow a specific convention and have three parts for position-specific n-grams and two parts otherwise. The parts are separated by `_`. The `.` symbol is used to separate elements within a part. The general naming scheme is POSITION_NGRAM_DISTANCE. The optional POSITION part of the name indicates the actual position of the n-gram in the sequence(s) and will be present only if `pos = TRUE`. This part is always a single integer. The NGRAM part of the name is a sequence of

elements in the n-gram. For example, 4.2.2 indicates the n-gram 422 (e.g. TCC). The DISTANCE part of the name is a vector of distance(s). For example, 0.0 indicates zero distances (continuous n-grams), while 1.2 represents distances for the n-gram A_A__A.

Examples of n-gram names:

- 46_4.4.4_0.1 : trigram 44_4 on position 46
- 12_2.1_2 : bigram 2__1 on position 12
- 8_1.1.1_0.0 : continuous trigram 111 on position 8
- 1.1.1_0.0 : continuous trigram 111 without position information

Value

a [simple_triplet_matrix](#) where columns represent n-grams and rows sequences. See Details for specifics of the naming convention.

Note

By default, the counted n-gram data is stored in a memory-saving format. To convert an object to a 'classical' matrix use the [as.matrix](#) function. See examples for further information.

See Also

Create vector of possible n-grams: [create_ngrams](#).

Extract n-grams from sequence(s): [seq2ngrams](#).

Get indices of n-grams: [get_ngrams_ind](#).

Count n-grams for multiple values of n: [count_multigrams](#).

Count only specified n-grams: [count_specified](#).

Examples

```
# count trigrams without position information for nucleotides
count_ngrams(sample(1L:4, 50, replace = TRUE), 3, 1L:4, pos = FALSE)
# count position-specific trigrams from multiple nucleotide sequences
seqs <- matrix(sample(1L:4, 600, replace = TRUE), ncol = 50)
ngrams <- count_ngrams(seqs, 3, 1L:4, pos = TRUE)
# output results of the n-gram counting to screen
as.matrix(ngrams)
```

count_total	<i>Count total number of n-grams</i>
-------------	--------------------------------------

Description

Computes total number of n-grams that can be extracted from sequences.

Usage

```
count_total(seq, n, d)
```

Arguments

seq	a vector or matrix describing sequence(s).
n	integer size of n-gram.
d	integer vector of distances between elements of n-gram (0 means consecutive elements). See Details.

Details

The maximum number of possible n-grams is limited by their length and the distance between elements of the n-gram.

Value

An integer rpresenting the total number of n-grams.

Note

A format of d vector is discussed in Details of [count_ngrams](#). The maximum

Examples

```
seqs <- matrix(sample(1L:4, 600, replace = TRUE), ncol = 50)
# make several sequences shorter by replacing them partially with NA
seqs[8L:11, 46L:50] <- NA
seqs[1L, 31L:50] <- NA
count_total(seqs, 3, c(1, 0))
```

create_encoding	<i>Create encoding</i>
-----------------	------------------------

Description

Reduces an alphabet using physicochemical properties.

Usage

```
create_encoding(prop, len)
```

Arguments

prop	matrix of properties with number of column equal to the length of the alphabet. Column must be named after elements of the alphabet. Each row represents a different physicochemical property.
len	length of the resulting encoding. Must be larger than zero and smaller than number of elements in the alphabet.

Details

The encoding is a list of groups to which elements of an alphabet should be reduced. All elements of the alphabet (all amino acids or all nucleotides) should appear in the encoding.

Value

An encoding.

See Also

[calc_ed](#): calculate the encoding distance between two encodings. [encoding2df](#): converts an encoding to a data frame. [validate_encoding](#): validate a structure of an encoding.

Examples

```
enc1 = list(`1` = c("a", "t"),
           `2` = c("g", "c"))
encoding2df(enc1)
```

create_feature_target *Create feature according to given contingency matrix*

Description

Creates a matrix of features and target based on the values from contingency matrix.

Usage

```
create_feature_target(n11, n01, n10, n00)
```

Arguments

n11	number of elements for which both target and feature equal 1.
n01	number of elements for which target and feature equal 1,0 respectively.
n10	number of elements for which target and feature equal 0,1 respectively.
n00	number of elements for which both target and feature equal 0.

Value

a matrix of 2 columns and $n11+n10+n01+n00$ rows. Columns represent target and feature vectors, respectively.

Examples

```
# equivalent of
#       target
# feature 10 375
#       15 600
target_feature <- create_feature_target(10, 375, 15, 600)
```

create_ngrams *Get all possible n-Grams*

Description

Creates the vector of all possible n_grams (for given n).

Usage

```
create_ngrams(n, u, possible_grams = NULL)
```


Arguments

`n` integer size of n-gram.
`u` integer, numeric or character vector of all possible unigrams.
`possible_grams` number of possible n-grams. If not NULL n-grams do not contain information about position

Details

See Details section of [count_ngrams](#) for more information about n-grams naming convention. The possible information about distance must be added by hand (see examples).

Value

a character vector. Elements of n-gram are separated by dot.

Note

Input data must be a matrix or data frame of numeric elements.

Examples

```
# bigrams for standard aminoacids
create_ngrams(2, 1L:20)
# bigrams for standard aminoacids with positions, 10 amino acid long sequence, so
# only 9 bigrams can be located in sequence
create_ngrams(2, 1L:20, 9)
# bigrams for DNA with positions, 10 nucleotide long sequence, distance 1, so only
# 8 bigrams in sequence
# paste0 adds information about distance at the end of n-gram
paste0(create_ngrams(2, 1L:4, 8), "_0")
```

```
criterion_distribution
```

```
      criterion_distribution class
```

Description

A result of [distr_crit](#) function.

Details

An object of class `criterion_distribution` is a numeric matrix.

Data

1st column: possible values of criterion.
2nd column: probability density function.
3rd column: cumulative distribution function.

Attributes

plot_data A matrix with values of the criterion and their probabilities.

nice_name 'Nice' name of the criterion.

cut.feature_test	<i>Categorize tested features</i>
------------------	-----------------------------------

Description

Categorizes results of `test_features` function into groups based on their significance.

Usage

```
## S3 method for class 'feature_test'
cut(x, split = "significances", breaks = c(0, 1e-04, 0.01, 0.05, 1), ...)
```

Arguments

x	an object of class <code>feature_test</code> .
split	attribute along which output should be categorized. Possible values are "significances", "positives" and "negatives". See Value.
breaks	a vector of significances of frequencies along which n-grams are aggregated. See description of <code>cut</code> function and Details.
...	further parameters accepted by the <code>cut</code> function.

Value

the value of function depends on the `split` parameter. The function returns a named list of length equal to the length of significances (when `split` equals "significances") or frequencies (when `split` equals "positives" or "negatives") minus one. Each elements of the list contains names of the n-grams belonging to the given significance or frequency group.

decode_ngrams	<i>Decode n-grams</i>
---------------	-----------------------

Description

Transforms a vector of n-grams into a human-friendly form.

Usage

```
decode_ngrams(ngrams)
```

Arguments

ngrams a character vector of n-grams.

Value

a character vector of length equal to the number of n-grams.

Note

Decoded n-grams lose the position information.

See Also

Validate n-gram structure: [is_ngram](#).

Inverse function: [code_ngrams](#).

Examples

```
decode_ngrams(c("2_1.1.2_0.1", "3_1.1.2_2.0", "3_2.2.2_0.0"))
```

degenerate

Degenerate protein sequence

Description

'Degenerates' amino acid or nucleic sequence by aggregating elements to bigger groups.

Usage

```
degenerate(seq, element_groups)
```

Arguments

seq character vector or matrix representing single sequence.

element_groups encoding of elements: list of groups to which elements of sequence should be aggregated. Must have unique names.

Value

A character vector or matrix (if input is a matrix) containing aggregated elements.

Note

Characters not present in the element_groups will be converted to NA with a warning.

See Also

[l2n](#) to easily convert information stored in biological sequences from letters to numbers. [calc_ed](#) to calculate distance between encodings.

Examples

```
sample_seq <- c(1, 3, 1, 3, 4, 4, 3, 1, 2)
table(sample_seq)

# aggregate sequence to purins and pyrimidines
deg_seq <- degenerate(sample_seq, list(w = c(1, 4), s = c(2, 3)))
table(deg_seq)
```

degenerate_ngrams	<i>Degenerate n-grams</i>
-------------------	---------------------------

Description

'Degenerates' n-grams by aggregating amino acid or nucleotide elements into bigger groups.

Usage

```
degenerate_ngrams(x, element_groups, binarize = FALSE)
```

Arguments

x	object containing n-grams.
element_groups	encoding of elements: list of groups to which elements of n-grams should be aggregated. Must have unique names.
binarize	logical indicating if n-grams should be binarized

Value

A character vector or matrix (if input is a matrix) containing degenerated n-grams.

distr_crit	<i>Compute criterion distribution</i>
------------	---------------------------------------

Description

Computes criterion distribution under null hypothesis for all contingency tables possible for a feature and a target.

Usage

```
distr_crit(target, feature, criterion = "ig", iter_limit = 200)
```

Arguments

target	{0,1}-valued target vector. See Details.
feature	{0,1}-valued feature vector. See Details.
criterion	criterion used for calculations of distribution. See calc_criterion for the list of available criteria.
iter_limit	limit the number of calculated contingency matrices. If NULL, computes all possible contingency matrices.

Details

both target and feature vectors may contain only 0 and 1.

Value

An object of class [criterion_distribution](#).

See Also

[calc_criterion](#).

Examples

```
target_feature <- create_feature_target(10, 375, 15, 600)
distr_crit(target = target_feature[,1], feature = target_feature[,2])
```

encoding2df *Convert encoding to data frame*

Description

Converts an encoding to a data frame.

Usage

```
encoding2df(x, sort = FALSE)
```

Arguments

x encoding.
sort if TRUE rows are sorted according to elements.

Details

The encoding is a list of groups to which elements of an alphabet should be reduced. All elements of the alphabet (all amino acids or all nucleotides) should appear in the encoding.

Value

data frame with two columns. First column represents an index of a group in the supplied encoding and the second column contains all elements of the encoding.

See Also

[calc_ed](#): calculate the encoding distance between two encodings. [encoding2df](#): converts an encoding to a data frame. [validate_encoding](#): validate a structure of an encoding.

Examples

```
create_encoding(aaprop[1L:5, ], 5)
```

fast_crosstable *2d cross-tabulation*

Description

Quickly cross-tabulates two binary vectors.

Usage

```
fast_crosstable(target, len_target, pos_target, feature)
```

Arguments

target	target.
len_target	length of the target vector.
pos_target	number of positive cases in the target vector.
feature	feature vector.

Details

Input looks odd, but the function was build to be fast subroutine of `calc_ig`, which works on many features but only one target.

Value

a vector of length four:

1. target +, feature+
2. target +, feature-
3. target -, feature+
4. target -, feature-

Note

Binary vector means a numeric vector with 0 or 1.

Examples

```
tar <- sample(0L:1, 100, replace = TRUE)
feat <- sample(0L:1, 100, replace = TRUE)
fast_crosstable(tar, length(tar), sum(tar), feat)
```

feature_test	<i>feature_test class</i>
--------------	---------------------------

Description

A result of `test_features` function.

Details

An object of the `feature_test` class is a numeric vector of p-values. Additional attributes characterizes further the details of test which returned these p-values.

Attributes

criterion the criterion used in permutation test.

adjust the name of p-value adjusting method.

times the number of permutations. If QuiPT was chosen NA.

occ frequency of features splitted in subset based on the value of target.

See Also

Methods:

- [as.data.frame.feature_test](#)
- [cut.feature_test](#)
- [print.feature_test](#)
- [summary.feature_test](#)

full2simple

Convert encoding from full to simple format

Description

Converts an encoding from the full format to the simple format.

Usage

```
full2simple(x)
```

Arguments

x encoding.

Examples

```
aa1 = list(`1` = c("g", "a", "p", "v", "m", "l", "i"),
           `2` = c("k", "h"),
           `3` = c("d", "e"),
           `4` = c("f", "r", "w", "y", "s", "t", "c", "n", "q"))
full2simple(aa1)
```

gap_ngrams

Gap n-grams

Description

Introduces gaps in the n-grams.

Usage

```
gap_ngrams(ngrams)
```

Arguments

ngrams a vector of positioned n-grams (as created by [count_ngrams](#)).

Details

A single element of the input n-gram at a time will be replaced by a gap. For example, introducing gaps in n-gram 2_1 . 1 . 2_0 . 1 will results in three n-grams: 3_1 . 2_1 (where the 2_1_0 unigram was replaced by a gap), 2_1 . 2_2 and 2_1 . 1_0.

Value

A character vector of (n-1)-grams with introduced gaps.

See Also

Reverse function: [add_1grams](#).

Examples

```
gap_ngrams(c("2_1.1.2_0.1", "3_1.1.2_0.0", "3_2.2.2_0.0"))
gap_ngrams(c("1.1.2_0.1", "1.1.2_0.0", "2.2.2_0.0"))
```

generate_sequence

Generate sequence

Description

Generate a sequences using an alphabet of unigrams and set of rules.

Usage

```
generate_sequence(alphabet, regions)
```

Arguments

alphabet	the unigram alphabet. Columns are equivalent to unigrams and rows to particular properties.
regions	a list of rules describing regions.

```
generate_single_region
```

Generate single region

Description

Generate a region using an alphabet of unigrams and considering provided set of rules.

Usage

```
generate_single_region(alphabet, reg_len, prop_ranges, exactness)
```

Arguments

alphabet	the unigram alphabet. Columns are equivalent to unigrams and rows to particular properties.
reg_len	the number of unigrams inside the region.
prop_ranges	required intervals of properties of unigrams in the region. See Details.
exactness	a numeric value between 0 and 1 defining how strictly unigrams are kept within prop_ranges. If 1, only unigrams within prop_ranges are inside the region. if 0.9, there is 10 unigrams that are not in the prop_ranges will be inside the region.

Examples

```
props1 <- list(P1 = c(0, 0.5),
              P2 = c(0.2, 0.4),
              P3 = c(0.5, 1),
              P4 = c(0, 0))
```

```
props2 <- list(P1 = c(0.5, 1),
              P2 = c(0.4, 1),
              P3 = c(0, 0.5),
              P4 = c(1, 1))
```

```
alph <- generate_unigrams(c(replicate(8, props1, simplify = FALSE),
                           replicate(12, props2, simplify = FALSE)),
                          unigram_names = letters[1L:20])
```

```
rules1 <- list(P1 = c(0.5, 1),
              P2 = c(0.4, 1),
```

```
P3 = c(0, 0.5),
P4 = c(1, 1))

generate_single_region(alph, 10, rules1, 0.9)
```

```
generate_single_unigram
    Generate single unigram
```

Description

Assign randomly generated properties to a single unigram.

Usage

```
generate_single_unigram(unigram_ranges)
```

Arguments

`unigram_ranges` list of ranges containing respective properties. If named, names are preserved.

See Also

`generate_single_unigram` is a helper function for [generate_unigrams](#).

Examples

```
generate_single_unigram(list(P1 = c(0, 0.5),
                             P2 = c(0.2, 0.4),
                             P3 = c(0.5, 1),
                             P4 = c(0, 0)))
```

```
generate_unigrams    Generate unigrams
```

Description

Generates an alphabet of unigrams based on given list of properties.

Usage

```
generate_unigrams(unigram_list, unigram_names = NULL, prop_names = NULL)
```

Arguments

unigram_list a list of unigrams' parameters. See Details.
 unigram_names names of unigrams. If not NULL, will overwrite any existing unigram names.
 prop_names names of properties. If not NULL, will overwrite any existing names.

Details

Unigram parameters are represented as a list of intervals, where each interval corresponds to a different property. The function generate unigrams randomly choosing values of properties from given intervals using uniform distribution. All lists of ranges should have the same length, which equals to describing each unigram using the same properties.

Examples

```
props1 <- list(P1 = c(0, 0.5),
              P2 = c(0.2, 0.4),
              P3 = c(0.5, 1),
              P4 = c(0, 0))

props2 <- list(P1 = c(0.5, 1),
              P2 = c(0.4, 1),
              P3 = c(0, 0.5),
              P4 = c(1, 1))

alph <- generate_unigrams(c(replicate(8, props1, simplify = FALSE),
                           replicate(12, props2, simplify = FALSE)),
                          unigram_names = letters[1L:20])
```

 get_ngrams_ind

Get indices of n-grams

Description

Computes list of n-gram elements positions in sequence.

Usage

```
get_ngrams_ind(len_seq, n, d)
```

Arguments

len_seq integer value describing sequence's length.
 n integer size of n-gram.
 d integer vector of distances between elements of n-gram (0 means consecutive elements). See Details.

Details

A format of d vector is discussed in Details of [count_ngrams](#).

Value

A list with number of elements equal to n. Every element is a vector containing locations of given n-gram letter. For example, first element of list contain indices of first letter of all n-grams. The attribute d of output contains distances between letter used to compute locations (see Details).

Examples

```
# positions trigrams in sequence of length 10
get_ngrams_ind(10, 9, 0)
```

human_cleave	<i>Human signal peptides cleavage sites</i>
--------------	---

Description

A set of 648 cleavage sites and 648 parts of mature proteins shortly after cleavage sites derived from human proteome.

Format

A data frame with 1296 observations on the following 10 variables. Columns from P1 to P9 describes positions in an extracted peptide. tar is a target vector. It has value 1 if a peptide is a cleavage site and 0 if not.

Details

Each peptide in the data set is nine amino acid residues long. In case of cleavage sites, the cleavage is located between fifth and sixth peptide. The non-cleavage sites are parts of mature proteins starting five positions after cleavage site.

Note

Amino acid residues were recoded as integers.

Source

[UniProt](#)

Examples

```
data(human_cleave)
table(human_cleave[, 1])
```

`is_ngram`*Validate n-gram*

Description

Checks if the character string may be used as an n-gram and its notation follows specific convention of biogram package.

Usage

```
is_ngram(x)
```

Arguments

`x` character string representing single n-gram.

Value

TRUE if n-gram's notation is correct, FALSE if not.

Examples

```
print(is_ngram("1_1.1.1_0.0"))  
print(is_ngram("not_ngram"))
```

`l2n`*Convert letters to numbers*

Description

Converts biological sequence from letter to number notation.

Usage

```
l2n(seq, seq_type)
```

Arguments

`seq` character vector or matrix representing single sequence.
`seq_type` the type of sequence. Can be rna, dna or prot.

Value

a numeric vector or matrix containing converted elements.

See Also

l2n is a wrapper around [degenerate](#).

Inverse function: [n2l](#).

Examples

```
sample_seq <- c("a", "d", "d", "g", "a", "g", "n", "a", "l")
l2n(sample_seq, "prot")
```

list2matrix	<i>Convert list of sequences to matrix</i>
-------------	--

Description

Converts list of sequences to matrix.

Usage

```
list2matrix(seq_list)
```

Arguments

seq_list list of sequences (e.g. as returned by the [read.fasta](#) function).

Value

A matrix with the number of rows equal to the number of sequences and the number of columns equal to the length of the longest sequence.

Note

Since matrix must have specified number of columns, ends of shorter sequences are completed with NAs.

Examples

```
list2matrix(list(s1 = c("c", "g", "g", "t"),
                 s2 = c("g", "t", "c", "t", "t", "g"),
                 s3 = c("a", "a", "t")))
```

n2l *Convert numbers to letters*

Description

Converts biological sequence from number to letter notation.

Usage

```
n2l(seq, seq_type)
```

Arguments

seq integer vector or matrix representing single sequence.
seq_type the type of sequence. Can be rna, dna or prot.

Value

a character vector or matrix containing converted elements.

See Also

n2l is a wrapper around [degenerate](#).

Inverse function: [l2n](#).

Examples

```
sample_seq <- c(1, 3, 3, 6, 1, 6, 12, 1, 10)  
n2l(sample_seq, "prot")
```

ngrams2df *n-grams to data frame*

Description

Transforms a vector of n-grams into a data frame.

Usage

```
ngrams2df(ngrams)
```

Arguments

ngrams a character vector of n-grams.

Value

a data.frame with 2 (in case of n-grams without known position) or three columns (n-grams with position information).

See Also

Decode n-grams: [decode_ngrams](#).

Examples

```
ngrams2df(c("2_1.1.2_0.0", "3_1.1.2_0.0", "3_2.2.2_0.0", "2_1.1_0"))
```

```
plot.criterion_distribution
```

Plot criterion distribution

Description

Plots results of [distr_crit](#) function.

Usage

```
## S3 method for class 'criterion_distribution'  
plot(x, ...)
```

Arguments

x object of class [criterion_distribution](#).
... further arguments passed to [plot](#).

Value

nothing.

Examples

```
target_feature <- create_feature_target(10, 375, 15, 600)  
example_result <- distr_crit(target = target_feature[,1],  
                             feature = target_feature[,2])  
  
plot(example_result)  
  
# a ggplot2 plot  
library(ggplot2)  
ggplot_distr <- function(x) {  
  b <- data.frame(cbind(x=as.numeric(rownames(attr(x, "plot_data"))),  
                      attr(x, "plot_data")))  
  d1 <- cbind(b[,c(1,2)], attr(x, "nice_name"))  
  d2 <- cbind(b[,c(1,3)], "Probability")  
}
```

```

colnames(d1) <- c("x", "y", "panel")
colnames(d2) <- c("x", "y", "panel")
d <- rbind(d1, d2)
p <- ggplot(data = d, mapping = aes(x = x, y = y)) +
  facet_grid(panel~., scale="free") +
  geom_freqpoly(data= d2, aes(color=y), stat = "identity") +
  scale_fill_brewer(palette = "Set1") +
  geom_point(data=d1, aes(size=y), stat = "identity") +
  guides(color = "none") +
  guides(size = "none") +
  xlab("Number of cases with feature=1 and target=1") + ylab("")
p
}
ggplot_distr(example_result)

```

position_ngrams

Position n-grams

Description

Transforms a vector of positioned n-grams into a list of positions filled with n-grams that start on them.

Usage

```
position_ngrams(ngrams, df = FALSE, unigrams_output = TRUE)
```

Arguments

ngrams	a vector of positioned n-grams (as created by count_ngrams).
df	logical, if TRUE returns a data frame, if FALSE returns a list.
unigrams_output	logical, if TRUE extracts unigrams from the data and returns information about their position.

Value

if df is FALSE, returns a list of length equal to the number of unique n-gram starts present in n-grams. Each element of the list contains n-grams that start on this position. If df is TRUE, returns a data frame where first column contains n-grams and the second column represent their start positions.

See Also

Transform n-gram name to human-friendly form: [decode_ngrams](#).

Validate n-gram structure: [is_ngram](#).

Examples

```
# position data in the list format
position_ngrams(c("2_1.1.2_0.1", "3_1.1.2_0.0", "3_2.2.2_0.0"))
# position data in the data frame format
position_ngrams(c("2_1.1.2_0.1", "3_1.1.2_0.0", "3_2.2.2_0.0"), df = TRUE)
```

print.feature_test	<i>Print tested features</i>
--------------------	------------------------------

Description

Prints results of [test_features](#) function.

Usage

```
## S3 method for class 'feature_test'
print(x, ...)
```

Arguments

x	object of class feature_test .
...	further arguments passed to print.default .

Value

nothing.

read_fasta	<i>Read FASTA files</i>
------------	-------------------------

Description

A lightweight tool to read nucleic or amino-acid sequences from a file in FASTA format.

Usage

```
read_fasta(file)
```

Arguments

file	the name of the file which the data are to be read from.
------	--

Value

a list of sequences.

See Also

[read.fasta](#): heavier function for processing FASTA files.

Examples

```
## Not run:  
read_fasta("https://www.uniprot.org/uniprot/P28307.fasta")  
  
## End(Not run)
```

regenerate

Regenerate n-grams

Description

'Regenerates' amino acid or nucleic sequence written in a simplified alphabet by converting groups to regular expression.

Usage

```
regenerate(x, element_groups)
```

Arguments

x character string representing single n-gram.
element_groups encoding of elements: list of groups to which elements of sequence should be aggregated. Must have unique names.

Value

A character string representing a POSIX regular expression.

Note

Gaps (_) will be converted to any possible character from the alphabet (nucleotides or amino acids).

See Also

[degenerate](#) to easily convert information stored in biological sequences from letters to numbers.
[calc_ed](#) to calculate distance between simplified alphabets.

Examples

```
regenerate("ssw", list(w = c(1, 4), s = c(2, 3)))
```

regional_param	<i>regional_param class</i>
----------------	-----------------------------

Description

List of rules defining the region.

Details

An object of the regional_param class is a list consisting of all rules necessary to properly build a region.

Attributes

reg_len the number of unigrams inside the region. Might be 0

prop_ranges required intervals of properties of unigrams in the region

exactness a numeric value between 0 and 1 defining how strictly unigrams are kept within prop_ranges. If 1, only unigrams within prop_ranges are inside the region. if 0.9, there is 10 unigrams that are not in the prop_ranges will be inside the region.

See Also

[generate_sequence](#)

seq2ngrams	<i>Extract n-grams from sequence</i>
------------	--------------------------------------

Description

Extracts vector of n-grams present in sequence(s).

Usage

```
seq2ngrams(seq, n, u, d = 0, pos = FALSE)
```

Arguments

seq	a vector or matrix describing sequence(s).
n	integer size of n-gram.
u	integer, numeric or character vector of all possible unigrams.
d	integer vector of distances between elements of n-gram (0 means consecutive elements). See Details.
pos	logical, if TRUE position-specific n_grams are counted.

Details

A format of d vector is discussed in Details of [count_ngrams](#).

Value

A character matrix of n-grams, where every row corresponds to a different sequence.

Examples

```
# trigrams from multiple sequences
seqs <- matrix(sample(1L:4, 600, replace = TRUE), ncol = 50)
seq2ngrams(seqs, 3, 1L:4)
```

simple2full

Convert encoding from simple to full format

Description

Converts an encoding from the simple format to the full format.

Usage

```
simple2full(x)
```

Arguments

x encoding (see Details).

Details

The encoding should be named. Each name should correspond to a different amino acid or nucleotide.

Examples

```
aa1 = structure(c("1", "4", "3", "3", "4", "1", "2", "1", "2", "1",
                 "1", "4", "1", "4", "4", "4", "4", "1", "4", "4"),
               .Names = c("a", "c", "d", "e", "f", "g", "h", "i",
                          "k", "l", "m", "n", "p", "q",
                          "r", "s", "t", "v", "w", "y"))
simple2full(aa1)
```

summary.feature_test *Summarize tested features*

Description

Summarizes results of `test_features` function.

Usage

```
## S3 method for class 'feature_test'
summary(object, conf_level = 0.95, ...)
```

Arguments

object	of class <code>feature_test</code> .
conf_level	confidence level. A feature with p-value equal to or smaller than the confidence is considered significant.
...	ignored

Value

nothing.

table_ngrams *Tabulate n-grams*

Description

Builds a contingency table of the n-gram counts versus their class labels.

Usage

```
table_ngrams(seq, ngrams, target)
```

Arguments

seq	vector or matrix describing sequence(s).
ngrams	vector of n-grams.
target	integer vector with target information (e.g. class labels). Must have at least two values.

Value

a data frame with the number of columns equal to the length of the target plus 1. The first column contains names of the n-grams. Further columns represents counts of n-grams for respective value of the target.

Examples

```

seqs_pos <- matrix(sample(c("a", "c", "g", "t"), 100, replace = TRUE,
  prob = c(0.2, 0.4, 0.35, 0.05)), ncol = 5)
seqs_neg <- matrix(sample(c("a", "c", "g", "t"), 100, replace = TRUE),
  ncol = 5)
tab <- table_ngrams(seq = rbind(seqs_pos, seqs_neg),
  ngrams = c("1_c.t_0", "1_g.g_0", "2_t.c_0", "2_g.g_0", "3_c.c_0", "3_g.c_0"),
  target = c(rep(1, 20), rep(0, 20)))
# see the results
print(tab)
# easily plot the results using ggplot2

```

test_features

Permutation test for feature selection

Description

Performs a feature selection on positioned n-gram data using a Fisher's permutation test.

Usage

```

test_features(
  target,
  features,
  criterion = "ig",
  adjust = "BH",
  threshold = 1,
  quick = TRUE,
  times = 1e+05
)

```

Arguments

target	integer vector with target information (e.g. class labels).
features	integer matrix of features with number of rows equal to the length of the target vector.
criterion	criterion used in permutation test. See Details for the list of possible criterions.
adjust	name of p-value adjustment method. See p.adjust for the list of possible values. If NULL, p-values are not adjusted.
threshold	integer. Features that occur less than threshold and more often than <code>nrow(features) - threshold</code> are discarded from the permutation test.
quick	logical, if TRUE Quick Permutation Test (QuiPT) is used. If FALSE, normal permutation test is performed.
times	number of times procedure should be repeated. Ignored if quick is TRUE.

Details

Since the procedure involves multiple testing, it is advisable to use one of the available p-value adjustment methods. Such methods can be used directly by specifying the `adjust` parameter.

Available criterions:

ig Information Gain: [calc_ig](#).

kl Kullback-Leibler divergence: [calc_kl](#).

cs Chi-squared-based measure: [calc_cs](#).

Value

an object of class `feature_test`.

Note

Both `target` and `features` must be binary, i.e. contain only 0 and 1 values.

Features occurring too often and too rarely are considered not informative and may be removed using the `threshold` parameter.

References

Radivojac P, Obradovic Z, Dunker AK, Vucetic S, *Feature selection filters based on the permutation test* in Machine Learning: ECML 2004, 15th European Conference on Machine Learning, Springer, 2004.

See Also

[binarize](#) - binarizes input data.

[calc_criterion](#) - computes selected criterion.

[distr_crit](#) - distribution of criterion used in QuiPT.

[summary.feature_test](#) - summary of results.

[cut.feature_test](#) - aggregates test results in groups based on feature's p-value.

Examples

```
# significant feature
tar_feat1 <- create_feature_target(10, 390, 0, 600)
# significant feature
tar_feat2 <- create_feature_target(9, 391, 1, 599)
# insignificant feature
tar_feat3 <- create_feature_target(198, 202, 300, 300)
test_res <- test_features(tar_feat1[, 1], cbind(tar_feat1[, 2], tar_feat2[, 2],
                                             tar_feat3[, 2]))

summary(test_res)
cut(test_res)

# real data example
# we will analyze only a subsample of a dataset to make analysis quicker
```

```
ids <- c(1L:100, 701L:800)
deg_seqs <- degenerate(human_cleave[ids, 1L:9],
  list(`a` = c(1, 6, 8, 10, 11, 18),
    `b` = c(2, 5, 13, 14, 16, 17, 19, 20),
    `c` = c(3, 4, 7, 9, 12, 15)))

# positioned n-grams example
bigrams_pos <- count_ngrams(deg_seqs, 2, letters[1L:3], pos = TRUE)
test_features(human_cleave[ids, 10], bigrams_pos)

# unpositioned n-grams example, binarization required
bigrams_notpos <- count_ngrams(deg_seqs, 2, letters[1L:3], pos = TRUE)
test_features(human_cleave[ids, 10], binarize(bigrams_notpos))
```

validate_encoding	<i>Validate encoding</i>
-------------------	--------------------------

Description

Checks the structure of an encoding.

Usage

```
validate_encoding(x, u)
```

Arguments

x	encoding.
u	integer, numeric or character vector of all elements belonging to the encoding. See Details.

Details

The encoding is a list of groups to which elements of an alphabet should be reduced. All elements of the alphabet (all amino acids or all nucleotides) should appear in the encoding.

Value

TRUE if the x is a correctly reduced u, FALSE in any other cases.

See Also

[calc_ed](#): calculate the encoding distance between two encodings. [encoding2df](#): converts an encoding to a data frame.

Examples

```

enc1 = list(`1` = c("a", "t"),
            `2` = c("g", "c"))
# see if enc1 is the correctly reduced nucleotide (DNA) alphabet
validate_encoding(enc1, c("a", "c", "g", "t"))

# enc1 is not the RNA alphabet, so the results is FALSE
validate_encoding(enc1, c("a", "c", "g", "u"))

# validate_encoding works also on other notations
enc2 = list(a = c(1, 4),
            b = c(2, 3))
validate_encoding(enc2, 1L:4)

```

write_encoding	<i>Write encodings to a file</i>
----------------	----------------------------------

Description

Saves a list of encodings (or a single encoding to the file).

Usage

```
write_encoding(x, file = "")
```

Arguments

x	encoding or list of encodings.
file	either a character string naming a file or a connection open for writing. "" indicates output to the console.

Examples

```

aa1 = list(`1` = c("g", "a", "p", "v", "m", "l", "i"),
           `2` = c("k", "h"),
           `3` = c("d", "e"),
           `4` = c("f", "r", "w", "y", "s", "t", "c", "n", "q"))
write_encoding(aa1)

```

write_fasta	<i>Write FASTA files</i>
-------------	--------------------------

Description

A lightweight tool to read nucleic or amino-acid sequences from a file in FASTA format.

Usage

```
write_fasta(seq, file, nchar = 80)
```

Arguments

seq	a list of sequences.
file	the name of the output file.
nchar	the number of characters per line.

See Also

[write.fasta](#): heavier function for writing FASTA files.

Index

- *Topic **datasets**
 - aaprop, 5
 - human_cleave, 53
- *Topic **distribution**
 - distr_crit, 45
- *Topic **hclust**
 - create_encoding, 39
- *Topic **manip**
 - cut.feature_test, 42
 - degenerate, 43
 - encoding2df, 46
 - l2n, 54
 - list2matrix, 55
 - n2l, 56
 - regenerate, 60
 - summary.feature_test, 63
 - validate_encoding, 66
- *Topic **nonparametric**
 - test_features, 64

- aaprop, 5
- add_1grams, 21, 49
- as.data.frame.feature_test, 22, 48
- as.matrix, 36

- binarize, 23, 65
- biogram (biogram-package), 3
- biogram-package, 3

- calc_criterion, 23, 24, 27, 30, 45, 65
- calc_cs, 24, 65
- calc_ed, 4, 25, 30, 39, 44, 46, 60, 66
- calc_ig, 26, 47, 65
- calc_kl, 27, 65
- calc_pi, 28
- calc_si, 25, 28, 29
- check_criterion, 30
- chisq.test, 25
- cluster_reg_exp, 31
- code_ngrams, 32, 43

- connection, 67
- construct_ngrams, 32
- count_multigrams, 34, 36
- count_ngrams, 3, 34, 35, 37, 38, 41, 49, 53, 58, 62
- count_specified, 36, 37, 37
- count_total, 38
- create_encoding, 39
- create_feature_target, 40
- create_ngrams, 36, 40
- criterion_distribution, 41, 45, 57
- cut, 42
- cut.feature_test, 42, 48, 65

- data.frame, 22
- decode_ngrams, 32, 42, 57, 58
- degenerate, 4, 43, 55, 56, 60
- degenerate_ngrams, 44
- distr_crit, 41, 45, 57, 65

- encoding2df, 25, 28, 39, 46, 46, 66

- fast_crosstable, 46
- feature_test, 22, 42, 47, 59, 63, 65
- full2simple, 48

- gap_ngrams, 22, 49
- generate_sequence, 49, 61
- generate_single_region, 50
- generate_single_unigram, 51
- generate_unigrams, 51, 51
- get_ngrams_ind, 36, 52

- human_cleave, 53

- is_ngram, 43, 54, 58

- KL.plugin, 28

- l2n, 44, 54, 56
- list2matrix, 55

n2l, [55](#), [56](#)
ngrams2df, [56](#)

p.adjust, [64](#)
plot, [57](#)
plot.criterion_distribution, [57](#)
position_ngrams, [58](#)
print.default, [59](#)
print.feature_test, [48](#), [59](#)

read.fasta, [55](#), [60](#)
read_fasta, [59](#)
regenerate, [60](#)
regional_param, [61](#)

seq2ngrams, [36](#), [61](#)
simple2full, [62](#)
simple_triplet_matrix, [23](#), [36](#), [37](#)
summary.feature_test, [48](#), [63](#), [65](#)

table_ngrams, [63](#)
test_features, [4](#), [22–25](#), [28](#), [33](#), [42](#), [47](#), [59](#),
[63](#), [64](#)

validate_encoding, [25](#), [28](#), [29](#), [39](#), [46](#), [66](#)

write.fasta, [68](#)
write_encoding, [67](#)
write_fasta, [68](#)