Package 'canprot'

May 8, 2026

Date 2024-03-28

Version 2.0.0

Title Chemical Analysis of Proteins

Maintainer Jeffrey Dick <j3ffdick@gmail.com>

Depends R (>= 3.1.0)

Imports stringi, multcompView

Suggests knitr, rmarkdown, tinytest, CHNOSZ (>= 1.3.4)

Description Chemical analysis of proteins based on their amino acid compositions. Amino acid compositions can be read from FASTA files and used to calculate chemical metrics including carbon oxidation state and stoichiometric hydration state, as described in Dick et al. (2020) <doi:10.5194/bg-17-6145-2020>. Other properties that can be calculated include protein length, grand average of hydropathy (GRAVY), isoelectric point (pI), molecular weight (MW), standard molal volume (V0), and metabolic costs (Akashi and Gojobori, 2002 <doi:10.1073/pnas.062526999>; Wagner, 2005 <doi:10.1093/molbev/msi126>; Zhang et al., 2018 <doi:10.1038/s41467-018-06461-1>). A database of amino acid compositions of human proteins derived from UniProt is provided.

Encoding UTF-8

License GPL-3

BuildResaveData no

VignetteBuilder knitr

URL https://github.com/jedick/canprot

NeedsCompilation no

Author Jeffrey Dick [aut, cre] (ORCID:
<https://orcid.org/0000-0002-0687-5890>)

Repository CRAN

Date/Publication 2024-03-28 16:00:13 UTC

Contents

`canprot-package` *Chemical analysis of proteins*

Description

Chemical metrics of proteins are important for understanding biomolecular adaptation to environments. **canprot** computes chemical metrics of proteins from their amino acid compositions.

Details

- `read_fasta` – start here to read amino acid compositions from FASTA files.
- `metrics` – use these functions to calculate chemical metrics from amino acid compositions.

Chemical metrics include carbon oxidation state (Z_C) stoichiometric oxidation and hydration state (n_{O_2} and n_{H_2O}) as described in Dick et al. (2020). Other variables that can be calculated are protein length, average molecular weight of amino acid residues, grand average of hydropathy (GRAVY), and isoelectric point (pI).

Differential expression datasets that were in the package up to version 1.1.2, mainly for the paper by Dick (2021), have been moved to JMDplots.

Three demos are available:

- `demo("thermophiles")`: Specific entropy vs Z_C for methanogen genomes and Nitrososphaeria MAGs. Amino acid compositions and optimal growth temperatures for methanogens were obtained from Dick et al. (2023). Accession numbers and data for Nitrososphaeria MAGs were taken from Tables S1 and S3 Luo et al. (2024) and are saved in '`extdata/aa/nitrososphaeria_MAGs.csv`'. Sequences of MAGs were downloaded from NCBI and processed to obtain amino acid compositions using the script in '`extdata/aa/prepare.R`'
- `demo("locations")`: Z_C and pI for human proteins with different subcellular locations. The localization data was taken from Thul et al. (2017) and is stored in '`extdata/protein/TAW+17_Table_S6_Validated.`
- `demo("redoxins")`: Z_C vs the midpoint reduction potentials of ferredoxin and thioredoxin in spinach and glutaredoxin and thioredoxin in *E. coli*. The sequences were obtained from UniProt and is stored in '`extdata/fasta/redoxin.fasta`'. The reduction potential data was taken from Åslund et al. (1997) and Hirasawa et al. (1999) and is stored in '`extdata/fasta/redoxin.csv`'. This file also lists UniProt IDs and start and stop positions for the protein chains excluding initiator methionines and signal peptides.

References

Åslund F, Berndt KD, Holmgren A. 1997. Redox potentials of glutaredoxins and other thiol-disulfide oxidoreductases of the thioredoxin superfamily determined by direct protein-protein redox equilibria. *Journal of Biological Chemistry* **272**(49): 30780–30786. doi:10.1074/jbc.272.49.30780

Dick JM. 2021. Water as a reactant in the differential expression of proteins in cancer. *Computational and Systems Oncology* **1**(1): e1007. doi:10.1002/cso2.1007

Dick JM, Yu M, Tan J. 2020. Uncovering chemical signatures of salinity gradients through compositional analysis of protein sequences. *Biogeosciences* **17**(23): 6145–6162. doi:10.5194/bg176145-2020

Dick JM, Boyer GM, Canovas PA, Shock EL. 2023. Using thermodynamics to obtain geochemical information from genomes. *Geobiology* **21**(2): 262–273. doi:10.1111/gbi.12532

Hirasawa M, Schürmann P, Jacquot J-P, Manieri W, Jacquot P, Keryer E, Hartman FC, Knaff DB. 1999. Oxidation-reduction properties of chloroplast thioredoxins, ferredoxin:thioredoxin reductase, and thioredoxin *f*-regulated enzymes. *Biochemistry* **38**(16): 5200–5205. doi:10.1021/bi982783v

Luo Z-H, Li Q, Xie Y-G, Lv A-P, Qi Y-L, Li M-M, Qu Y-N, Liu Z-T, Li Y-X, Rao Y-Z, et al. 2024 Jan. Temperature, pH, and oxygen availability contributed to the functional differentiation of ancient *Nitrososphaeria*. *The ISME Journal* **18**(1): wrad031. doi:10.1093/ismejo/wrad031

Thul PJ, Åkesson L, Wiking M, Mahdessian D, Geladaki A, Blal HA, Alm T, Asplund A, Björk L, Breckels LM, et al. 2017. A subcellular map of the human proteome. *Science* **356**(6340): eaal3321. doi:10.1126/science.aal3321

`add_cld` *Compact letter display*

Description

Adds compact letter display (significant difference letters) to a boxplot.

Usage

```
add_cld(datlist, bp, dx = NULL, dy = NULL)
```

Arguments

`datlist`	list, list of data values for different groups
`bp`	list, output of `boxplot`
`dx`	numeric, offset for letters in the x direction
`dy`	numeric, offset for letters in the y direction

Details

This function adds a compact letter display (cld) to an existing boxplot. It calculates a one-way ANOVA with `aov` followed by Tukey's Honest Significant Differences with `TukeyHSD`, then obtains the cld with `multcompLetters4`. The letters are added to the plot at the upper right sides of the bars. Default values for `dx` and `dy` are computed from the current plot dimensions; these values can be adjusted if needed.

Value

Invisibly returns a list with dx, dy, and letters (letters used for the cld, in the same order as the groups in datlist).

Examples

```
# Are there significant differences of nH2O among human proteins with different Zc?
aa <- get("human.aa", canprot)
# Remove extremely short sequences
aa <- aa[!plength(aa) < 20, ]
Zc <- Zc(aa)
ilo <- Zc < -0.15
ihi <- Zc > -0.10
imid <- !ilo & !ihi
nH2O <- nH2O(aa)
nH2Olist <- list(lo.Zc = nH2O[ilo], mid.Zc = nH2O[imid], hi.Zc = nH2O[ihi])
bp <- boxplot(nH2Olist, ylab = cplab$nH2O)
add_cld(nH2Olist, bp)
# Yes, higher Zc is associated with lower nH2O
```

---

add_hull *Add convex hull*

Description

Adds a convex hull around the data.

Usage

```
add_hull(x, y = NULL, ...)
```

Arguments

x	x values
y	y values
...	arguments for polygon

Details

add_hull draws a convex hull around the points given in x, y. This function is a wrapper for chull and polygon.

Value

Invisibly returns the result from chull.

Examples

```
dat <- iris[, 1:2]
plot(dat)
add_hull(dat)
```

`calc_metrics` *Calculate one or more chemical metrics*

Description

Calculates selected chemical metrics from amino acid composition(s) of proteins.

Usage

```
calc_metrics(AAcomp, metrics = c("Zc", "nO2", "nH2O"), ...)
```

Arguments

`AAcomp`	data frame with amino acid compositions
`metrics`	character, chemical metrics to calculate
`...`	additional arguments passed to individual functions

Details

This is a wrapper function for the functions described at `metrics`. This wrapper implements case-insensitve matching to the function names (e.g., `zc` matches `Zc`). The following additional shortcuts are defined (e.g., `length` can be used in place of the actual function name, `plength`):

`length`	`plength`
`H/C`, `H_C`	`HC`
`N/C`, `N_C`	`NC`
`O/C`, `O_C`	`OC`
`S/C`, `S_C`	`SC`

Value

A data frame with the same number of rows as `AAcomp` and one column of numeric values for each of the `metrics`. An error is produced if any of the `metrics` is not available for calculation.

See Also

`metrics`

Examples

```
# Define the acid composition of alanylglycine
AG <- data.frame(Ala = 1, Gly = 1)
# Calculate default metrics (Zc, nO2, nH2O)
calc_metrics(AG)
# Calculate selected metrics
calc_metrics(AG, c("H/C", "O/C", "Length"))
```

CLES *Common language effect size*

Description

Calculate the common language effect size.

Usage

```
CLES(x, y, distribution = "normal")
```

Arguments

x	numeric, data
y	numeric, data
distribution	'normal' to use probabilities calculated for a normal distribution, or NA for empirical probabilities

Details

The common language statistic is defined for continuous data as "the probability that a score sampled at random from one distribution will be greater than a score sampled from some other distribution" (McGraw and Wong, 1992).

Given the default value of `distribution` ('normal'), this function uses pnorm to calculate the probability that a random sample from the unit normal distribution is greater than the Z score (i.e. (the mean of 'y' minus the mean of 'x') / square root of (variance of 'x' plus variance of 'y')).

If `distribution` is NA, this function calculates the empirical probability that the difference is positive, that is, the fraction of all possible pairings between elements of x and y where the difference ('y' value - 'x' value) is positive. It may not be possible to calculate the empirical probability for very large samples because of memory limits.

The examples use *simulated data for normal distributions*, given the sample size, mean, and standard deviation of datasets cited by McGraw and Wong, 1992. Therefore, the empirical probability in the examples approaches the normal curve probability. However, the empirical probability for *nonnormal* distributions is distinct from the normal curve probability, as discussed on p. 364-365 of McGraw and Wong, 1992.

References

McGraw KO, Wong SP. 1992. A common language effect size statistic. *Psychological Bulletin* **111**(2): 361–365. doi:10.1037/00332909.111.2.361

National Center for Health Statistics. 1987. Anthropometric Reference Data and Prevalence of Overweight: United States, 1976-1980. U.S. Department of Health and Human Services. Data from the National Health Survey, Series 11, No. 238.

Examples

```
# Example 1: Height differences between males and females
# a) Use statistics quoted by McGraw and Wong, 1992 from NCHS, 1987
# for heights in inches of 18-24 year-old males and females
# Table 14: number, mean height, and standard deviation of height of females
n1 <- 1066
M1 <- 64.3
SD1 <- 2.8
# Table 13: number, mean height, and standard deviation of height of males
n2 <- 988
M2 <- 69.7
SD2 <- 2.6
# b) Simulate data from a normal distribution with exact mean and SD
# use rnorm2 function from Ben Bolker's answer to
# https://stackoverflow.com/questions/18919091/generate-random-numbers-with-fixed-mean-and-sd
rnorm2 <- function(n, mean, sd) { mean + sd * scale(rnorm(n)) }
set.seed(1234)
height_female <- rnorm2(n1, M1, SD1)
height_male <- rnorm2(n2, M2, SD2)
# c) Calculate the CLES using the normal distribution and empirical probability
CLES_normal <- CLES(height_female, height_male)
CLES_empirical <- CLES(height_female, height_male, distribution = NA)
# d) Test numerical equivalence of the results
# The CLES is approximately 0.92 (McGraw and Wong, 1992)
# (note: becasue we used rnorm2, this doesn't depend on the seed)
stopifnot(all.equal(CLES_normal, 0.92, tol = 0.01))
# With this seed, the difference between the normal curve probability
# and empirical probability is less than 1%
stopifnot(all.equal(CLES_normal, CLES_empirical, tol = 0.01))

# Example 1.5: Use multiple simulated datasets to show approach
# of empirical probability to normal curve probability
CLES_empirical_n <- sapply(1:100, function(x) {
  height_female <- rnorm2(n1, M1, SD1)
  height_male <- rnorm2(n2, M2, SD2)
  CLES(height_female, height_male, distribution = NA)
})
CLES_empirical <- mean(CLES_empirical_n)
# now we're even closer to the normal curve probability
stopifnot(all.equal(CLES_normal, CLES_empirical, tol = 0.0001))

# Example 2: Multiple datasets in Table 2 of McGraw and Wong, 1992
# Sample statistics for females
```

```
n1 <- c(638, 672, 3139, 420740, 19274, 104263, 207, 394, 1066, 982, 108, 108)
M1 <- c(103, 15, 103, 18.9, 30, 16.1, 6.9, 13.3, 64.3, 134, 45, 94)
SD1 <- sqrt(c(908, 74, 219, 27, 110, 59, 15, 164, 6.8, 688, 310, 1971))
# Sample statistics for males
n2 <- c(354, 359, 3028, 356704, 21768, 133882, 199, 469, 988, 988, 443, 443)
M2 <- c(112, 23, 100, 17.9, 33, 18.6, 9.3, 21.8, 69.7, 163, 86, 212)
SD2 <- sqrt(c(1096, 96, 202, 29, 110, 61, 15, 133, 7.8, 784, 818, 5852))
# A function to calculate the effect size using simulated data
CLESfun <- function(n1, M1, SD1, n2, M2, SD2, distribution) {
  rnorm2 <- function(n, mean, sd) { mean + sd * scale(rnorm(n)) }
  set.seed(1234)
  x <- rnorm2(n1, M1, SD1)
  y <- rnorm2(n2, M2, SD2)
  CLES(x, y, distribution)
}
# Calculate 100 * CL for the normal curve probabilities
CLnorm <- sapply(1:12, function(i) {
  CL <- CLESfun(n1[i], M1[i], SD1[i], n2[i], M2[i], SD2[i], "normal")
  round(100 * CL)
})
# Calculate 100 * CL for empirical probabilities
CLemp <- sapply(1:12, function(i) {
  # skip very large samples: not enough memory
  if(n1[i] > 5000 | n2[i] > 5000) NA else {
    CL <- CLESfun(n1[i], M1[i], SD1[i], n2[i], M2[i], SD2[i], NA)
    round(100 * CL)
  }
})
# The difference between the empirical and normal curve
# probabilities is not more than 1 percent
stopifnot(max(abs(CLemp - CLnorm), na.rm = TRUE) <= 1)
# TODO: Why are some of the calculated values different from
# Table 2 of McGraw and Wong, 1992?
CLref <- c(54, 74, 44, 45, 56, 63, 67, 65, 92, 78, 89, 91)
# Differences range from -4 to 4
range(CLnorm - CLref)
#stopifnot(max(abs(CLnorm - CLref)) == 0)
```

---

human *Amino acid compositions of human proteins*

Description

Amino acid compositions of human proteins derived from UniProt.

Format

human.aa is a data frame with 25 columns in the format used for amino acid compositions in **CHNOSZ** (see thermo):

protein	character	Identification of protein
organism	character	Identification of organism
ref	character	Reference key for source of sequence data
abbrv	character	Abbreviation or other ID for protein (e.g. gene name)
chains	numeric	Number of polypeptide chains in the protein
Ala...Tyr	numeric	Number of each amino acid in the protein

The protein column contains UniProt IDs in the format database|accession-isoform, where database is most often 'sp' (Swiss-Prot) or 'tr' (TrEMBL), and isoform is an optional suffix indicating the isoform of the protein (particularly in the human.additional file).

Details

The amino acid compositions of human proteins are stored in three files under extdata/protein.

- human.base.rds contains amino acid compositions of canonical isoforms of manually reviewed proteins in the UniProt reference human proteome (computed from sequences in UP000005640_9606.fasta.gz, dated 2016-04-03).
- human.additional.rds contains amino acid compositions of additional proteins (UP000005640_9606_additional.fasta.gz) including isoforms and unreviewed sequences. In version 0.1.5, this file was trimmed to include only those proteins that are used in any of the datasets in the package.
- human.extra.csv contains amino acid compositions of other ("extra") proteins used in a dataset but not listed in one of the files above. These proteins may include obsolete, unreviewed, or newer additions to the UniProt database. Most, but not all, sequences here are HUMAN (see the organism column and the ref column for the reference keys).

On loading the package, the individual data files are read and combined, and the result is assigned to the human.aa object in the canprot environment.

See Also

human_aa gets amino acid compositions for human proteins specified by their UniProt IDs.

Examples

```
# The number of proteins
nrow(get("human.aa", canprot))
```

---

human_aa *Get amino acid compositions of human proteins*

Description

Get amino acid compositions of human proteins from their UniProt IDs.

Usage

```
human_aa(uniprot = NULL, aa_file = NULL,
  stop_if_missing = FALSE, warn_if_duplicated = FALSE)
```

Arguments

uniprot	character, UniProt IDs of proteins
aa_file	character, file name
stop_if_missing	logical, stop with an error if there are UniProt IDs that can't be found?
warn_if_duplicated	logical, emit a warning if duplicate UniProt IDs are detected?

Details

This function retrieves the amino acid compositions of one or more proteins specified by `uniprot`.

This function depends on the amino acid compositions of human proteins, which are stored in the `canprot` environment when the package is attached. If `aa_file` is specified, additional amino acid compositions are read from this file. This file should be in the same format as `human.extra.csv` in the installation directory of the package.

Value

The function returns a data frame with amino acid compositions of proteins.

Examples

```
human_aa("P24298")
```

---

metrics *Calculate chemical metrics for proteins*

Description

Calculate chemical metrics for proteins from their amino acid compositions.

Usage

```
Zc(AAcomp, ...)
nO2(AAcomp, basis = "QEC", ...)
nH2O(AAcomp, basis = "QEC", terminal_H2O = 0)
GRAVY(AAcomp, ...)
pI(AAcomp, terminal_H2O = 1, ...)
MW(AAcomp, terminal_H2O = 0, ...)
pMW(AAcomp, terminal_H2O = 1, ...)
V0(AAcomp, terminal_H2O = 0, ...)
```

```
pV0(AAcomp, terminal_H2O = 1, ...)
V0g(AAcomp, ...)
Density(AAcomp, ...)
S0(AAcomp, terminal_H2O = 0, ...)
pS0(AAcomp, terminal_H2O = 1, ...)
S0g(AAcomp, ...)
SV(AAcomp, ...)
Zcg(AAcomp, ...)
nH2Og(AAcomp, ...)
nO2g(AAcomp, ...)
HC(AAcomp, ...)
NC(AAcomp, ...)
OC(AAcomp, ...)
SC(AAcomp, ...)
nC(AAcomp, ...)
pnC(AAcomp, ...)
plength(AAcomp, ...)
Cost(AAcomp, ...)
RespiratoryCost(AAcomp, ...)
FermentativeCost(AAcomp, ...)
B20Cost(AAcomp, ...)
Y20Cost(AAcomp, ...)
H11Cost(AAcomp, ...)
cplab
```

Arguments

AAcomp	data frame, amino acid compositions
...	ignored additional arguments
basis	character, set of basis species
terminal_H2O	numeric, number of pairs of terminal groups

Details

Columns in AAcomp should be named with the three-letter abbreviations for the amino acids. Case-insensitive matching matching of the abbreviations is used; e.g., 'Ala', 'ALA', 'ala' all refer to alanine.

Metrics are normalized per amino acid residue except for Zc, pI, Density, plength, and other functions starting with p (for protein). The contribution of protein terminal groups (-H and -OH) to residue-normalized metrics is turned off by default. Set terminal_H2O to 1 (or to the number of polypeptide chains, if greater than one) to include their contribution.

The metrics are described below:

Zc Average oxidation state of carbon (Z_C) (Dick, 2014). This metric is independent of the choice of basis species. Note that Z_C is normalized by number of carbon atoms, not by number of residues.

nO2 Stoichiometric oxidation state (n_{O_2} per residue). The available basis species are:

- 'QEC' - glutamine, glutamic acid, cysteine, H_2O, O_2 (Dick et al., 2020)
- 'QCa' - glutamine, cysteine, acetic acid, H_2O, O_2

nH2O Stoichiometric hydration state (n_{H_2O} per residue). The basis species also affect this calculation.

GRAVY Grand average of hydropathy. Values of the hydropathy index for individual amino acids are from Kyte and Doolittle (1982).

pI Isoelectric point. The net charge for each ionizable group was pre-calculated from pH 0 to 14 at intervals of 0.01. The isoelectric point is found as the pH where the sum of charges of all groups in the protein is closest to zero. The p*K* values for the terminal groups and sidechains are taken from Bjellqvist et al. (1993) and Bjellqvist et al. (1994); note that the calculation does not implement position-specific adjustments described in the latter paper. The number of N- and C-terminal groups is taken from `terminal_H2O`.

MW Molecular weight.

pMW Molecular weight per protein.

V0 Standard molal volume. The values are derived from group contributions of amino acid sidechains and protein backbones (Dick et al., 2006).

pV0 Standard molal volume per protein.

V0g Specific volume (reciprocal density).

Density Density (MW / V0).

S0 Standard molal entropy. The values are derived from group contributions of amino acid sidechains and protein backbones (Dick et al., 2006).

pS0 Standard molal entropy per protein.

S0g Specific entropy.

SV Entropy density.

Zcg Carbon oxidation state per gram.

nO2g Stoichiometric oxidation state per gram.

nH2Og Stoichiometric hydration state per gram.

HC H/C ratio (not counting terminal -H and -OH groups).

NC N/C ratio.

OC O/C ratio (not counting terminal -H and -OH groups).

SC S/C ratio.

nC Number of carbon atoms per residue.

pnC Number of carbon atoms per protein.

plength Protein length (number of amino acid residues).

Cost Metabolic cost (Akashi and Gojobori, 2002).

RespiratoryCost Respiratory cost (Wagner, 2005).

FermentativeCost Fermentative cost (Wagner, 2005).

B20Cost Biosynthetic cost in bacteria (Zhang et al., 2018).

Y20Cost Biosynthetic cost in yeast (Zhang et al., 2018).

`H11Cost` Biosynthetic cost in humans (Zhang et al., 2018).

`...` is provided to permit `get` or `do.call` constructions with the same arguments for all metrics. For instance, a `terminal_H2O` argument can be suppled to either `Zc` or `nH2O`, but it only has an effect on the latter.

`cplab` is a list of formatted labels for each of the chemical metrics listed here. A check in the code ensures that the names of the functions for calculating metrics and the names for labels listed `cplab` are identical.

References

Akashi H, Gojobori T. 2002. Metabolic efficiency and amino acid composition in the proteomes of *Escherichia coli* and *Bacillus subtilis*. *Proceedings of the National Academy of Sciences* **99**(6): 3695–3700. doi:10.1073/pnas.062526999

Bjellqvist B, Hughes GJ, Pasquali C, Paquet N, Ravier F, Sanchez J-C, Frutiger S, Hochstrasser D. 1993. The focusing positions of polypeptides in immobilized pH gradients can be predicted from their amino acid sequences. *Electrophoresis* **14**: 1023–1031. doi:10.1002/elps.11501401163

Bjellqvist B, Basse B, Olsen E, Celis JE. 1994. Reference points for comparisons of two-dimensional maps of proteins from different human cell types defined in a pH scale where isoelectric points correlate with polypeptide compositions. *Electrophoresis* **15**: 529–539. doi:10.1002/elps.1150150171

Dick JM, LaRowe DE, Helgeson HC. 2006. Temperature, pressure, and electrochemical constraints on protein speciation: Group additivity calculation of the standard molal thermodynamic properties of ionized unfolded proteins. *Biogeosciences* **3**(3): 311–336. doi:10.5194/bg33112006

Dick JM. 2014. Average oxidation state of carbon in proteins. *J. R. Soc. Interface* **11**: 20131095. doi:10.1098/rsif.2013.1095

Dick JM, Yu M, Tan J. 2020. Uncovering chemical signatures of salinity gradients through compositional analysis of protein sequences. *Biogeosciences* **17**: 6145–6162. doi:10.5194/bg1761452020

Kyte J, Doolittle RF. 1982. A simple method for displaying the hydropathic character of a protein. *J. Mol. Biol.* **157**: 105–132. doi:10.1016/00222836(82)905150

Wagner A. 2005. Energy constraints on the evolution of gene expression. *Molecular Biology and Evolution* **22**(6): 1365–1374. doi:10.1093/molbev/msi126

Zhang H, Wang Y, Li J, Chen H, He X, Zhang H, Liang H, Lu J. 2018. Biosynthetic energy cost for amino acids decreases in cancer evolution. *Nature Communications* **9**(1): 4124. doi:10.1038/s41467018064611

See Also

For calculation of Z_C from an elemental formula (instead of amino acid composition), see the `ZC` function in **CHNOSZ**. `calc_metrics` is a wrapper to calculate one or more metrics specified in an argument.

Examples

```
# Amino acid composition of a tripeptide (Gly-Ala-Gly)
aa <- data.frame(Ala = 1, Gly = 2)
# Calculate Zc, nH2O, and length
Zc(aa)
```

```
nH2O(aa)
plength(aa)

# Make a plot with formatted labels
plot(Zc(aa), nH2O(aa), xlab = cplab$Zc, ylab = cplab$nH2O)
```

`read_fasta` *Functions for reading FASTA files*

Description

Read protein amino acid composition or sequences from a file and count numbers of amino acids in given sequences.

Usage

```
read_fasta(file, iseq = NULL, type = "count", lines = NULL,
  ihead = NULL, start = NULL, stop = NULL, molecule = "protein", id = NULL)
count_aa(sequence, start = NULL, stop = NULL, molecule = "protein")
sum_aa(AAcomp, abundance = 1, average = FALSE)
```

Arguments

`file`	character, path to FASTA file
`iseq`	numeric, which sequences to read from the file
`type`	character, type of return value (‘`count`’, ‘`sequence`’, ‘`lines`’, or ‘`headers`’)
`lines`	list of character, supply the lines here instead of reading them from file
`ihead`	numeric, which lines are headers
`start`	numeric, position in sequence to start counting
`stop`	numeric, position in sequence to stop counting
`molecule`	character, type of molecule (‘`protein`’, ‘`DNA`’, or ‘`RNA`’)
`id`	character, value to be used for `protein` in output table
`sequence`	character, one or more sequences
`AAcomp`	data frame, amino acid composition(s) of proteins
`abundance`	numeric, abundances of proteins
`average`	logical, return the weighted average of amino acid counts?

Details

`read_fasta` is used to retrieve entries from a FASTA file. Use `iseq` to select the sequences to read (the default is all sequences).

The function returns various data formats depending on the value of `type`:

‘`count`’ data frame of amino acid counts

'sequence' list of sequences

'lines' list of lines from the FASTA file (including headers)

'headers' list of header lines from the FASTA file

When type is 'count', the header lines of the file are parsed to obtain protein names that are put into the protein column in the result. Furthermore, if a UniProt FASTA header is detected (using the regular expression "\|.....\|.*_"), the information there (accession, name, organism) is split into the protein, abbrv, and organism columns of the resulting data frame. this behavior (which may take a while for large files) can be suppressed by supplying protein names in id.

To speed up processing, if the line numbers of the header lines were previously determined, they can be supplied in ihead. Optionally, the lines of a previously read file may be supplied in lines (in this case no file is needed so file should be set to "").

count_aa is the underlying function that counts the numbers of each amino acid or nucleic-acid base in one or more sequences. The matching of letters is case-insensitive. A message is generated if any character in sequence, excluding spaces, is not one of the single-letter amino acid or nucleobase abbreviations. start and/or stop can be provided to process a fragment of the sequence. If only one of start or stop is present, the other defaults to 1 (start) or the length of the respective sequence (stop).

sum_aa sums the amino acid compositions in the input AAcomp data frame. It only applies to columns with the three-letter abbreviations of amino acids and to a column named chains (if present). The values in these columns are multiplied by the indicated abundance after recycling to the number of proteins. The values in these columns are then summed; if average is TRUE then the sum is divided by the number of proteins. Proteins with missing values (NA) of amino acid composition or abundance are omitted from the calculation. The output has one row and the same number of columns as the input; the value in the non-amino acid columns is taken from the first row of the input.

Value

count_aa returns a data frame with these columns (for proteins): Ala, Cys, Asp, Glu, Phe, Gly, His, Ile, Lys, Leu, Met, Asn, Pro, Gln, Arg, Ser, Thr, Val, Trp, Tyr. For 'DNA', the columns are changed to A, C, G, T, and for 'RNA', the columns are changed to A, C, G, U.

read_fasta returns a list of sequences (for type equal to 'sequence') or a list of lines (for type equal to 'lines' or 'headers'). Otherwise, (for type equal to 'count') a data frame with these columns: protein, organism, ref, abbrv, chains, and the columns described above for count_aa.

sum_aa returns a one-row data frame.

See Also

Pass the output of read_fasta to add.protein in the CHNOSZ package to set up thermodynamic calculations for proteins.

Examples

```
## Reading a protein FASTA file
# The path to the file
file <- system.file("extdata/fasta/KHAB17.fasta", package = "canprot")
# Read the sequences, and print the first one
```

```
read_fasta(file, type = "seq")[[1]]
# Count the amino acids in the sequences
aa <- read_fasta(file)
# Calculate protein length (number of amino acids in each protein)
plength(aa)
# Sum the amino acid compositions
sum_aa(aa)

# Count amino acids in a sequence
count_aa("GGSGG")
# A message is issued for unrecognized characters
count_aa("AAAXXX")
# Count nucleobases in a sequence
bases <- count_aa("ACCGGGTTT", molecule = "DNA")
```

Index