Package 'canprot'

October 12, 2022

Date 2022-01-17

Version 1.1.2

Title Chemical Metrics of Differentially Expressed Proteins

Maintainer Jeffrey Dick <j3ffdick@gmail.com>

Depends R (>= 3.1.0), CHNOSZ (>= 1.3.2)

Imports xtable, MASS, rmarkdown

Suggests knitr, testthat

Description Chemical metrics of differentially expressed proteins in cancer and cell culture proteomics experiments. Data files in the package have amino acid compositions of proteins obtained from UniProt and >250 published lists of up- and down-regulated proteins in different cancer types and laboratory experiments. Functions are provided to calculate chemical metrics including protein length, grand average of hydropathicity (GRAVY), isoelectric point (pI), carbon oxidation state, and stoichiometric hydration state; the latter two are described in Dick et al. (2020) <doi:10.5194/bg-17-6145-2020>. The vignettes visualize differences of chemical metrics between up- and down-regulated proteins and list literature references for all datasets.

Encoding UTF-8

License GPL (>= 2)

BuildResaveData no

VignetteBuilder knitr

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

NeedsCompilation no

Author Jeffrey Dick [aut, cre] (<https://orcid.org/0000-0002-0687-5890>), Ben Bolker [ctb] (<https://orcid.org/0000-0002-2127-0443>)

Repository CRAN

Date/Publication 2022-01-17 08:22:51 UTC

R topics documented:

`canprot-package` *Chemical Metrics of Differentially Expressed Proteins*

Description

canprot is a package for computing chemical metrics of proteins from their amino acid compositions. The package has datasets for differentially expressed proteins in cancer and cell culture conditions from over 250 studies.

Overview

This package includes datasets for differential expression of proteins in six cancer types (breast, colorectal, liver, lung, pancreatic, prostate), and four cell culture conditions (hypoxia, hyperosmotic stress, secreted proteins in hypoxia, and 3D compared to 2D growth conditions). The hyperosmotic stress data are divided into bacteria, archaea (both high- and low-salt experiments) and eukaryotes; the latter are further divided into salt and glucose experiments. Nearly all datasets use UniProt IDs; if not given in the original publications they have been added using the UniProt mapping tool (`https://www.uniprot.org/mapping/`).

The analysis vignettes have plots for each cancer type and cell culture condition and references for all data sources used. Because of their size, pre-built vignette HTML files are not included with the package; use `mkvig` to compile and view any of the vignettes.

The functions in this package were originally based on code for the papers of Dick (2016 and 2017). Updated data compilations and revised vignettes were developed by Dick et al. (2020) and Dick (2021).

References

Dick, J. M. (2016) Proteomic indicators of oxidation and hydration state in colorectal cancer. *PeerJ* **4**, e2238. doi: 10.7717/peerj.2238

Dick, J. M. (2017) Chemical composition and the potential for proteomic transformation in cancer, hypoxia, and hyperosmotic stress. *PeerJ* **5**, e3421. doi: 10.7717/peerj.3421

Dick, J. M., Yu, M. and Tan, J. (2020) Uncovering chemical signatures of salinity gradients through compositional analysis of protein sequences. *Biogeosciences* **17**, 6145–6162. doi: 10.5194/bg17-61452020

Dick, J. M. (2021) Water as a reactant in the differential expression of proteins in cancer. *Comp. Sys. Onco.* **1**:e1007. doi: 10.1002/cso2.1007

Examples

```
# List the data files for all studies
# (one study can have more than one dataset)
exprdata <- system.file("extdata/expression", package="canprot")
datafiles <- dir(exprdata, recursive=TRUE)
print(datafiles)
# Show the number of data files for each condition
table(dirname(datafiles))
```

`check_IDs` *Check UniProt IDs*

Description

Find the first ID for each protein that matches a known UniProt ID.

Usage

```
check_IDs(dat, IDcol, aa_file = NULL, updates_file = NULL)
```

Arguments

`dat`	data frame, protein expression data
`IDcol`	character, name of column that has the UniProt IDs
`aa_file`	character, name of file with additional amino acid compositions
`updates_file`	character, name of file with old to new ID mappings

Details

`check_IDs` is used to check for known UniProt IDs and to update obsolete IDs. The source IDs should be provided in the `IDcol` column of `dat`; multiple IDs for one protein can be separated by a semicolon.

The function keeps the first "known" ID for each protein, which must be present in one of these groups:

- The human_aa dataset of amino acid compositions.
- Old UniProt IDs that are mapped to new UniProt IDs in uniprot_updates or in updates_file if specified.
- IDs of proteins in aa_file, which lists amino acid compositions in the format described for human_aa (see extdata/protein/human_extra.csv for an example and thermo$protein for more details).

Value

dat is returned with possibly changed values in the column designated by IDcol; old IDs are replaced with new ones, the first known ID for each protein is kept, then proteins with no known IDs are assigned NA.

See Also

This function is used by the pdat_ functions, where it is called before cleanup.

Examples

```
# Make up some data for this example
ID <- c("P61247;PXXXXX", "PYYYYY;P46777;P60174", "PZZZZZ")
dat <- data.frame(ID = ID, stringsAsFactors = FALSE)
# Get the first known ID for each protein; the third one is NA
check_IDs(dat, "ID")

# Update an old ID
dat <- data.frame(Entry = "P50224", stringsAsFactors = FALSE)
check_IDs(dat, "Entry")
```

---

cleanup *Clean Up Data*

Description

Remove proteins with unavailable IDs, ambiguous expression ratios, and duplicated IDs.

Usage

```
cleanup(dat, IDcol, up2 = NULL)
```

Arguments

dat	data frame, protein expression data
IDcol	character, name of column that has the UniProt IDs
up2	logical, TRUE for up-regulated proteins, FALSE for down-regulated proteins

Details

cleanup is used in the pdat_ functions to clean up the dataset given in dat. IDcol is the name of the column that has the UniProt IDs, and up2 indicates the expression change for each protein. The function removes proteins with unavailable (NA or "") or duplicated IDs. If up2 is provided, the function also removes unquantified proteins (those that have NA values of up2) and those with ambiguous expression ratios (up and down for the same ID). For each operation, a message is printed describing the number of proteins that are 'unavailable', 'unquantified', 'ambiguous', or 'duplicated'.

Alternatively, if IDcol is a logical value, it selects proteins to be unconditionally removed.

See Also

This function is used extensively by the pdat_ functions, where it is called after check_IDs (if needed).

Examples

```
# Set up a simple workflow
extdatadir <- system.file("extdata", package="canprot")
datadir <- paste0(extdatadir, "/expression/pancreatic/")
dataset <- "CYD+05"
dat <- read.csv(paste0(datadir, dataset, ".csv.xz"), as.is = TRUE)
up2 <- dat$Ratio..cancer.normal. > 1
# Remove two unavailable and one duplicated proteins
dat <- cleanup(dat, "Entry", up2)
# Now we can retrieve the amino acid compositions
pcomp <- protcomp(dat$Entry)

# Read another data file
datadir <- paste0(system.file("extdata", package="canprot"), "/expression/colorectal/")
dataset <- "STK+15"
dat <- read.csv(paste0(datadir, "STK+15.csv.xz"), as.is = TRUE)
# Remove unavailable proteins
dat <- cleanup(dat, "uniprot")
# Remove proteins that have less than 2-fold expression ratio
dat <- cleanup(dat, abs(log2(dat$invratio)) < 1)
```

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, Kenneth O. and Wong, S. P. (1992) A common language effect size statistic. *Psychological Bulletin* **11**, 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.* Data from the National Health Survey, Series 11, No. 238. DHHS Publication (PHS) No. 87-1688. U.S. Government Printing Office, Washington, DC. https://www.cdc.gov/nchs/data/series/sr_11/sr11_238.pdf

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)
```

`diffplot` *Plot Differences of Chemical Metrics*

Description

Make a plot showing differences of selected chemical metrics.

Usage

```
diffplot(comptab, vars = c("ZC", "nH2O"), col = "black", plot.rect = FALSE,
         pt.text = c(letters, LETTERS), cex.text = 0.85, oldstyle = FALSE,
         pch = 1, cex = 2.1, contour = TRUE, col.contour = par("fg"),
         probs = 0.5, add = FALSE, labtext = NULL, ...)
cplab
```

Arguments

`comptab`	list or data frame, chemical differences generated by `get_comptab`
`vars`	character, which variables to plot
`col`	character or numeric, color(s) for the points
`plot.rect`	logical, plot a reference rectangle?
`pt.text`	character, text labels for the points
`cex.text`	numeric, size of text labels
`oldstyle`	logical, use old style plot?
`pch`	numeric, point symbol
`cex`	numeric, point size
`contour`	logical, add contour lines?
`col.contour`	character or numeric, color of contour lines
`probs`	numeric, probability level(s) for contours
`add`	logical, add to an existing plot?
`labtext`	character, text to add to axis labels
`...`	other argumenents passed to `plot`

Details

A plot is created with points showing the differences between up- and down-regulated proteins for two chemical metrics, as calculated by get_comptab. The default setting of vars refers to average oxidation state of carbon (Z_C) as the x-variable and stoichiometric hydration state (n_{H_2O}) as the y-variable.

The colors of the points are controlled by col, which is recycled to be equal to the number of comparisons in comptab.

If plot.rect is TRUE, a shaded rectangle is drawn with coordinates -0.01, -0.01, 0.01, 0.01. This is useful for visualizing the different scales of multi-panel plots.

If pt.text is not NA or FALSE, text labels are added with size controlled by cex.text. The default value produces labels that are taken sequentially from the 26 lowercase Roman letters in alphabetical order (letters), followed by the set of uppercase letters (LETTERS).

For labtext = NULL, descriptive text ("median difference" or "mean difference") is added to the axis labels in parentheses. This text can be changed by giving a value in labtext (for both axes), two values (for each axis), or NA to suppress the text.

cplab is a list of formatted labels used by diffplot. It is an exported object, available to the user and other packages.

Plot style

The overall style of the plot is controlled by oldstyle.

oldstyle = FALSE This is the current default style. Use pch and cex to control the point symbol and size. Contours are added for confidence regions of highest probability density, computed using a 2-D kernel density estimate (kde2d). probs gives the probability level(s) and col.contour sets the color(s) of the contour lines. contour can be a logical vector, indicating which points to include; set it to FALSE to omit the contour lines.

The code to calculate the contour levels is modified from HPDregionplot in the **emdbook** package by Ben Bolker (https://cran.r-project.org/package=emdbook).

oldstyle = TRUE This style was used for the historical (2017) vignettes, which have been moved to the 'extdata/cpcp' directory in **JMDplots** (https://github.com/jedick/JMDplots). For each dataset, the point symbol is a filled square if the *p*-values of both the x-variable and y-variable are less than 0.05, a filled circle if the *p*-value of one of the x- or y-variables is less than 0.05, and an open circle otherwise. A solid line is drawn from the point to the corresponding axis if the rounded, absolute value of (CLES in percent - 50) of the x- or y-variable is greater than or equal 10. Otherwise, a dashed line is drawn from the point to the corresponding axis if the *p*-value of the x- or y-variable is less than 0.05. Otherwise, no line is drawn.

See Also

qdist to plot quantile distributions for a single dataset.

Examples

```
library(CHNOSZ)
# Make an old-style plot for two datasets
```

```
comptab <- lapply(c("JKMF10", "WDO+15_C.N"), function(dataset) {
  pdat <- pdat_colorectal(dataset)
  get_comptab(pdat, oldstyle = TRUE)
})
diffplot(comptab, oldstyle = TRUE)
```

`get_comptab` *Calculate Differences of Chemical Metrics*

Description

Compute differences of carbon oxidation state, stoichiometric hydration state and other chemical metrics between groups of up- and down-regulated proteins.

Usage

```
get_comptab(pdat, var1 = "ZC", var2 = "nH2O", plot.it = FALSE,
  mfun = "median", oldstyle = FALSE, basis = getOption("basis"))
```

Arguments

`pdat`	list, data object generated by a `pdat_` function
`var1`	character, the first variable
`var2`	character, the second variable
`plot.it`	logical, make a scatterplot?
`mfun`	character, either '`median`' or '`mean`'
`oldstyle`	logical, also calculate `CLES` and *p*-values?
`basis`	character, keyword for basis species to use

Details

The available variables are:

'ZC'	average oxidation state of carbon (Z_C; see `ZCAA`)
'nH2O'	stoichiometric hydration state per residue (n_{H_2O}; see `H2OAA`)
'nO2'	stoichiometric oxidation state per residue (n_{O_2}; see `O2AA`)
'V0'	standard molal volume per residue
'nAA'	protein length (number of amino acids)
'GRAVY'	grand average of hydropathicity (see `GRAVY`)
'pI'	isoelectric point (see `pI`)
'MW'	molecular weight per residue

Differentially expressed proteins are identified by the value of `pdat$up2` (TRUE for up-regulated proteins and FALSE for down-regulated proteins). The differences are calculated as (median for up-

regulated proteins) - (median for down-regulated proteins); if mfun is 'mean', means of the groups are used instead. If oldstyle is TRUE, the function also calculates the common language effect size (CLES, in percent) and *p*-value for each variable.

The basis argument is used to select the basis species, which are used for the calculation of n_{H_2O} and n_{O_2}. The default for getOption("basis") is to use the 'QEC' basis species (see metrics).

Volume is calculated using amino acid group additivity as described by Dick et al. (2006).

Set plot.it to TRUE to make a scatterplot. Open red squares and filled blue circles stand for up-regulated and down-regulated proteins, respectively.

Value

A data frame is returned invisibly containing the columns 'dataset', 'description', 'n1' (number of down-regulated proteins), 'n2' (number of up-regulated proteins), followed two sets of columns for the variables. These are denoted generically as ('var.mfun1', 'var.mfun2', 'var.diff', 'var.CLES', 'var.p.value'), where 'var' is replaced by the name of var1 or var2, and 'mfun' is replaced by the value of mfun. For example, 'ZC.median1' and 'ZC.median2' are the median Z_C of the down- and up-regulated proteins, respectively.

References

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

Examples

```
pd <- pdat_colorectal("JKMF10")
# default variables: ZC and nH2O
get_comptab(pd, plot.it = TRUE)
# protein length and per-residue volume
get_comptab(pd, "nAA", "V0", plot.it = TRUE)
```

---

human *Amino Acid Compositions of Human Proteins*

Description

Data for amino acid compositions of proteins and conversion from old to new UniProt IDs.

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 human environment.

As an aid for processing datasets that list old (obsolete) UniProt IDs, the corresponding new (current) IDs are are stored in uniprot_updates. These ID mappings have been manually added as needed for individual datasets, and include proteins from humans as well as other organisms. check_IDs performs the conversion of old to new IDs.

See Also

Amino acid compositions of non-human proteins are stored under extdata/aa in directories archaea, bacteria, cow, dog, mouse, rat, and yeast. These files can be loaded in protcomp via the aa_file argument, which is used e.g. in pdat_osmotic_bact.

Examples

```
# The number of proteins
nrow(get("human_aa", human))
# The number of old to new ID mappings
nrow(get("uniprot_updates", human))
```

`metrics` *Calculate Chemical Metrics for Proteins*

Description

These functions calculate chemical metrics of proteins given a data frame of amino acid compositions.

Usage

```
ZCAA(AAcomp, nothing = NULL)
H2OAA(AAcomp, basis = getOption("basis"))
O2AA(AAcomp, basis = getOption("basis"))
GRAVY(AAcomp)
pI(AAcomp)
MWAA(AAcomp)
basis.text(basis)
```

Arguments

`AAcomp`	data frame, amino acid compositions
`nothing`	dummy argument
`basis`	character, basis species

Details

Columns in `AAcomp` should be named with the three-letter abbreviations for the amino acids ('`Ala`', '`Arg`', ...). Abbreviations are matched without regard to case (e.g. '`ALA`' is the same as '`ala`').

The metrics are described below:

`ZCAA` Average oxidation state of carbon (Z_C) (Dick, 2014). `nothing` is an extra argument that does nothing. It is provided so that `do.call` can be used to run `ZCAA` or `H2OAA` with the same number of arguments.

This metric is independent of the choice of basis species.

`H2OAA` Stoichiometric hydration state (n_{H_2O}) per residue. The available `basis` species are:

- '`QEC`' - glutamine, glutamic acid, cysteine, H_2O, O_2 (Dick et al., 2020) (this is the default for `getOption("basis")`)
- '`QCa`' - glutamine, cysteine, acetic acid, H_2O, O_2
- Any other valid basis specification for `basis`, such as '`CHNOS`' for CO_2, NH_3, H_2S, H_2O, and O_2

`O2AA` Stoichiometric oxidation state (n_{O_2}) per residue. The basis species also affect this calculation.

`GRAVY` Grand average of hydropathicity. 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 to be one, unless a value for `chains` (number of polypeptide chains) is given in `AAcomp`.

`MWAA` Molecular weight per residue.

Note that Z_C is a per-carbon average, but n_{H_2O} is a per-residue average. The contribution of H_2O from the terminal groups of proteins is counted, so shorter proteins have slightly greater n_{H_2O}.

Tests for a few proteins (see examples) indicate that `GRAVY` and `pI` are equal those calculated with the ProtParam tool (`https://web.expasy.org/protparam/`; Gasteiger et al., 2005).

`basis.text` is used in the vignettes to generate a textual description of the names of the basis species, except H_2O and O_2, for one of the keywords '`QEC`' or '`QCa`'.

References

Bjellqvist, B., Hughes, G. J., Pasquali, C., Paquet, N., Ravier, F., Sanchez, J.-C., Frutiger, S. and 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. and Basse, B. and Olsen, E. and Celis, J. E. (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, J. M. (2014) Average oxidation state of carbon in proteins. *J. R. Soc. Interface* **11**, 20131095. doi: 10.1098/rsif.2013.1095

Dick, J. M., Yu, M. and Tan, J. (2020) Uncovering chemical signatures of salinity gradients through compositional analysis of protein sequences. *Biogeosciences* **17**, 6145–6162. doi: 10.5194/bg17-61452020

Gasteiger, E., Hoogland, C., Gattiker, A., Duvaud, S., Wilkins, M. R., Appel, R. D. and Bairoch, A. (2005) Protein identification and analysis tools on the ExPASy server. In J. M. Walker (Ed.), *The Proteomics Protocols Handbook* (pp. 571–607). Totowa, NJ: Humana Press Inc. doi: 10.1385/1592598900:571

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

See Also

For calculation of Z_C from a chemical formula instead of amino acid composition, see the `ZC` function in **CHNOSZ**.

Examples

```
# we need CHNOSZ for these examples
require(CHNOSZ)
```

```
# for reference, compute ZC of alanine and glycine "by hand"
ZC.Gly <- ZC("C2H5NO2")
ZC.Ala <- ZC("C3H7NO2")
# define the composition of a Gly-Ala-Gly tripeptide
AAcomp <- data.frame(Gly = 2, Ala = 1)
# calculate the ZC of the tripeptide (value: 0.571)
ZC.GAG <- ZCAA(AAcomp)
# this is equal to the carbon-number-weighted average of the amino acids
nC.Gly <- 2 * 2
nC.Ala <- 1 * 3
ZC.average <- (nC.Gly * ZC.Gly + nC.Ala * ZC.Ala) / (nC.Ala + nC.Gly)
stopifnot(all.equal(ZC.GAG, ZC.average))

# compute the per-residue nH2O of Gly-Ala-Gly
basis("QEC")
nH2O.GAG <- species("Gly-Ala-Gly")$H2O
# divide by the length to get residue average (we keep the terminal H-OH)
nH2O.residue <- nH2O.GAG / 3
# compare with the value calculated by H2OAA() (-0.2)
nH2O.H2OAA <- H2OAA(AAcomp, "QEC")
stopifnot(all.equal(nH2O.residue, nH2O.H2OAA))

# calculate GRAVY for a few proteins
# first get the protein index in CHNOSZ's list of proteins
iprotein <- pinfo(c("LYSC_CHICK", "RNAS1_BOVIN", "AMYA_PYRFU"))
# then get the amino acid compositions
AAcomp <- pinfo(iprotein)
# then calculate GRAVY
Gcalc <- as.numeric(GRAVY(AAcomp))
# these are equal to values obtained with ProtParam on uniprot.org
# https://web.expasy.org/cgi-bin/protparam/protparam1?P00698@19-147@
# https://web.expasy.org/cgi-bin/protparam/protparam1?P61823@27-150@
# https://web.expasy.org/cgi-bin/protparam/protparam1?P49067@2-649@
Gref <- c(-0.472, -0.663, -0.325)
stopifnot(all.equal(round(Gcalc, 3), Gref))
# also calculate molecular weight of the proteins
MWcalc <- as.numeric(MWAA(AAcomp)) * protein.length(iprotein)
MWref <- c(14313.14, 13690.29, 76178.25)
stopifnot(all.equal(round(MWcalc, 2), MWref))

# calculate pI for a few proteins
iprotein <- pinfo(c("LYSC_CHICK", "RNAS1_BOVIN", "AMYA_PYRFU", "CSG_HALJP"))
AAcomp <- pinfo(iprotein)
pI_calc <- pI(AAcomp)
# reference values calculated with ProtParam on uniprot.org
# LYSC_CHICK: residues 19-147 (sequence v1)
# RNAS1_BOVIN: residues 27-150 (sequence v1)
# AMYA_PYRFU: residues 2-649 (sequence v2)
# CSG_HALJP: residues 35-862 (sequence v1)
pI_ref <- c(9.32, 8.64, 5.46, 3.37)
stopifnot(all.equal(as.numeric(pI_calc), pI_ref))
```

mkvig *Compile and View Vignettes from the Command Line*

Description

Compile the indicated vignette and open it in the browser.

Usage

```
mkvig(vig = NULL)
```

Arguments

vig	character, name of a vignette without '.Rmd' extension

Details

In order to reduce package space and check time, pre-built vignettes are not included in the package. mkvig is a convenience function to compile the vignettes on demand and view them in a browser.

The available vignettes for mkvig are listed here:

- *Cell culture* – 'hypoxia', 'secreted', 'osmotic_bact', 'osmotic_euk', 'osmotic_halo', 'glucose', '3D'
- *Cancer* – 'breast', 'colorectal', 'liver', 'lung', 'pancreatic', 'prostate'

Note that pandoc (including pandoc-citeproc), as a system dependency of **rmarkdown**, is required to build the vignettes.

See Also

The vignettes can also be run using e.g. demo("glucose"), and through the interactive help system (help.start > Packages > canprot > Code demos).

Examples

```
## Not run:
mkvig("colorectal")

## End(Not run)
```

pdat_ *Get Protein Expression Data*

Description

Get differentially expressed proteins and amino acid compositions.

Usage

```
pdat_breast(dataset = 2020)
pdat_colorectal(dataset = 2020)
pdat_liver(dataset = 2020)
pdat_lung(dataset = 2020)
pdat_pancreatic(dataset = 2020)
pdat_prostate(dataset = 2020)
pdat_hypoxia(dataset = 2020)
pdat_secreted(dataset = 2020)
pdat_3D(dataset = 2020)
pdat_glucose(dataset = 2020)
pdat_osmotic_bact(dataset = 2020)
pdat_osmotic_euk(dataset = 2020)
pdat_osmotic_halo(dataset = 2020)
.pdat_multi(dataset = 2020)
.pdat_osmotic(dataset = 2017)
```

Arguments

dataset	character, dataset name

Details

The pdat_ functions assemble lists of up- and down-regulated proteins and retrieve their amino acid compositions using protcomp. The result can be used with get_comptab to make a table of chemical metrics that can then be plotted with diffplot.

If dataset is '2020' (the default) or '2017', the function returns the names of all datasets in the compilation for the respective year.

Each dataset name starts with a reference key indicating the study (i.e. paper or other publication) where the data were reported. The reference keys are made by combining the first characters of the authors' family names with the 2-digit year of publication.

If a study has more than one dataset, the reference key is followed by an underscore and an identifier for the particular dataset. This identifier is saved in the variable named stage in the functions, but can be any descriptive text.

To retrieve the data, provide a single dataset name in the dataset argument. Protein expression data is read from the CSV files stored in extdata/expression/, under the subdirectory corresponding to the name of the pdat_ function. Some of the functions also read amino acid compositions (e.g. for non-human proteins) from the files in extdata/aa/.

Descriptions for each function:

- pdat_colorectal, pdat_pancreatic, pdat_breast, pdat_lung, pdat_prostate, and pdat_liver retrieve data for protein expression in different cancer types.
- pdat_hypoxia gets data for cellular extracts in hypoxia and pdat_secreted gets data for secreted proteins (e.g. exosomes) in hypoxia.
- pdat_3D retrieves data for 3D (e.g. tumor spheroids and aggregates) compared to 2D (monolayer) cell culture.
- .pdat_osmotic retrieves data for hyperosmotic stress, for the 2017 compilation only. In 2020, this compilation was expanded and split into pdat_osmotic_bact (bacteria), pdat_osmotic_euk (eukaryotic cells) and pdat_osmotic_halo (halophilic bacteria and archaea).
- pdat_glucose gets data for high-glucose experiments in eukaryotic cells.
- .pdat_multi retrieves data for studies that have multiple types of datasets (e.g. both cellular and secreted proteins in hypoxia), and is used internally by the specific functions (e.g. pdat_hypoxia and pdat_secreted).

Value

A list consisting of:

dataset Name of the dataset

description Descriptive text for the dataset, used for making the tables in the vignettes (see mkvig)

pcomp UniProt IDs together with amino acid compositions obtained using protcomp

up2 Logical vector with length equal to the number of proteins; TRUE for up-regulated proteins and FALSE for down-regulated proteins

Examples

```
# List datasets in the 2017 complilation for colorectal cancer
pdat_colorectal(2017)
# Get proteins and amino acid compositions for one dataset
pdat_colorectal("JKMF10")
```

---

protcomp *Amino Acid Compositions*

Description

Get amino acid compositions of proteins.

Usage

```
protcomp(uniprot = NULL, aa = NULL, aa_file = NULL)
```

Arguments

uniprot	character, UniProt IDs of proteins
aa	data frame, amino acid compositions
aa_file	character, file name

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 human environment when the package is attached. If aa_file is specified, additional amino acid compositions to be considered are read from this file, which should be in the same format as e.g. human_extra.csv (see also thermo$protein). Alternatively, the amino acid compositions can be given in aa, bypassing the search step.

Value

The function returns a list with elements uniprot (UniProt IDs as given in the arguments) and aa (amino acid compositions of the proteins).

See Also

cleanup

Examples

```
protcomp("P24298")
```

---

qdist *Quantile Distributions for One Dataset*

Description

Make a plot showing quantile distributions for up- and down-regulated proteins.

Usage

```
qdist(pdat, vars = c("ZC", "nH2O"), show.steps = FALSE)
```

Arguments

pdat	list, output of a pdat_ function for a single dataset
vars	character, which variables to plot
show.steps	logical, show the steps using plot.ecdf?

Details

This function makes a quantile distribution plot with lines for both up- and down-regulated proteins. The variable (`var`) can be 'ZC', 'H2O', or both (two plots are made for the latter). The horizontal axis is the variable and the vertical axis is the quantile point. A solid black line is drawn for the down-regulated proteins, and a dashed red line for the up-regulated proteins. The median difference is shown by a gray horizontal line drawn between the distributions at the 0.5 quantile point.

References

Jimenez, C. R. and Knol, J. C. and Meijer, G. A. and Fijneman, R. J. A. (2010) Proteomics of colorectal cancer: Overview of discovery studies and identification of commonly identified cancer-associated proteins and candidate CRC serum markers. *J. Proteomics* **73**, 1873–1895. doi: 10.1016/j.jprot.2010.06.004

See Also

`diffplot` to plot median differences for multiple datasets.

Examples

```
# Plot the data of Jimenez et al., 2010 for colorectal cancer
pdat <- pdat_colorectal("JKMF10")
qdist()
```

`xsummary` *Summarize Chemical Differences*

Description

Make an HTML table summarizing chemical differences.

Usage

```
xsummary(comptab, vars = c("ZC", "nH2O"))
xsummary2(comptab1, comptab2)
xsummary3(comptab1, comptab2, comptab3)
```

Arguments

`comptab`	list or data frame, summary of comparisons generated by `get_comptab`
`vars`	character, two variables to tabulate
`comptab1`	list, output of `get_comptab`
`comptab2`	list, output of `get_comptab`
`comptab3`	list, output of `get_comptab`

Details

xsummary makes an HTML table (using xtable) and adds bold and underline formatting to highlight significant chemical differences. The *p*-value is bolded if it is less than 0.05, and the percent common language effect size (CLES) is bolded if it is <= 40 or >= 60. The mean (or median) difference is [underlined / bolded] if [only one of / both] the *p*-value and CLES pass these cutoffs.

The generated table is written to the console, and can be used in a vignette using the results = "asis" chunk option. The function also returns (invisibly) the data frame used to make the table; this data frame differs from comptab by having row names added (alphabetical one-letter IDs for the datasets).

xsummary2 is an updated version that is used in the current vignettes in the package. It shows negative numbers in bold (*p*-value and CLES are not shown). xsummary3 is a further revision that shows GRAVY and pI; it is used in the 'osmotic_bact' and 'osmotic_halo' vignettes.

Examples

```
comptab <- lapply(c("JKMF10", "WDO+15_C.N"), function(dataset) {
  pdat <- pdat_colorectal(dataset)
  get_comptab(pdat, oldstyle = TRUE)
})
xsummary(comptab)
```

Index