Package ‘canprot’

February 26, 2019

Date 2019-02-26

Version 0.1.2

Title Chemical Composition of Differential Protein Expression

Author Jeffrey Dick

Maintainer Jeffrey Dick <j3ffdick@gmail.com>

Depends R (>= 3.1.0)

Imports CHNOSZ (>= 1.3.0), xtable

Suggests knitr, rmarkdown, KernSmooth

Description Datasets are collected here for differentially (up- and down-) expressed proteins identified in proteomic studies of cancer and in cell culture experiments. Tables of amino acid compositions of proteins are used for calculations of chemical composition, projected into selected basis species. Plotting functions are used to visualize the compositional differences and thermodynamic potentials for proteomic transformations.

License GPL-3

BuildResaveData no

VignetteBuilder knitr

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

NeedsCompilation no

Repository CRAN

Date/Publication 2019-02-26 21:40:08 UTC

R topics documented:

canprot-package *Differential Expression of Proteins in Cancer*

Description

canprot is a package for exploration of compositional data of proteomes and thermodynamic analysis of proteomic transformations using the concepts of chemical components (basis species) and chemical affinity (negative of Gibbs energy).

Overview

This package includes datasets for differential expression of proteins in colorectal cancer (CRC), pancreatic cancer, hypoxia, and hyperosmotic stress. All datasets use UniProt IDs, which have been added if not present in the original publications. The sources of data are listed in the vignettes and are further described in Dick (2016) and (2017). The functions in this package were derived from the code included as supplemental information for those studies.

References

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

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

Examples

```
# list all of the data files for protein expression
exprdata <- system.file("extdata/expression", package="canprot")
exprfiles <- dir(exprdata, recursive=TRUE)
print(exprfiles)
# get the reference keys from the filenames
refkeys <- gsub(".csv", "", sapply(strsplit(exprfiles, "/"), "[", 2))
# find the reference keys in the UniProt updates file
uu <- get("uniprot_updates", canprot)
update_keys <- unique(unlist(strsplit(uu$source, ";")))
```

```
# find the reference keys in the extra human amino acid composition file
he <- get("human_extra", canprot)
extra_keys <- unique(unlist(strsplit(he$ref, ";")))
# list the unused keys (these should be empty when the package is released)
setdiff(update_keys, refkeys)
setdiff(extra_keys, refkeys)
```

`canprot` *Amino Acid Compositions of Human Proteins*

Description

Data for amino acid compositions, and updates to UniProt IDs.

Format

The columns of `human_aa` are compatible with the layout 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 compositional data
`abbrv`	character	Abbreviation or other ID for protein
`chains`	numeric	Number of polypeptide chains in the protein
`Ala...Tyr`	numeric	Number of each amino acid in the protein

Here, the `protein` column contains the UniProt ID (accession), possibly with a suffix indicating the isoform of the protein (esp. from `human_additional.csv`).

Details

These amino acid compositions were compiled from amino acid sequences downloaded from UniProt. Amino acid sequences of human proteins were obtained from files in the UniProt reference proteome, dated 2016-04-03, downloaded from `ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/reference_proteomes/Eukaryota/`.

The amino acid compositions of human proteins are stored in three files. `human_base.Rdata` contains amino acid compositions of proteins in the UniProt reference proteome (`UP000005640_9606.fasta.gz` containing canonical, manually reviewed sequences). `human_additional.Rdata` contains amino acid compositions of additional proteins in the UniProt reference proteome (`UP000005640_9606_additional.fasta.gz` containing isoforms and unreviewed sequences). `human_extra.csv` contains amino acid compositions of other (“extra”) proteins identified in proteomic experiments but not listed in one of the files above.

On loading the package, the individual data files are read and combined using `rbind`, and the result is assigned to the `human_aa` object in the `canprot` environment.

As an aid for processing some datasets that use old (obsoleted) UniProt IDs, the corresponding new (current) IDs are are stored in `uniprot_updates`. `uniprot_updates` also lists the source (i.e. reference key) that uses each old ID.

Examples

```
nrow(get("human_aa", canprot))
```

`check_ID` *Check UniProt IDs*

Description

Do the IDs correspond to known UniProt IDs?

Usage

```
check_ID(ID, aa_file = NULL, updates_file = NULL)
```

Arguments

`ID`	character or list, candidate UniProt IDs
`aa_file`	character, file name
`updates_file`	character, file name

Details

`check_ID` is used to check candidate `ID`s supplied in a character or list object. Multiple IDs can be separated by a semicolon. "Known" IDs are those that are present in the `human_aa` dataset of amino acid compositions.

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`). If `updates_file` is specified, these ID mappings are included alongside the packaged `uniprot_updates`.

Value

The function returns the IDs in a list (dropping the semicolons, but reflecting the original arrangement), replacing any unknown IDs with `NA`.

Examples

```
# the function replaces the 3 non-existent IDs with NA
check_ID(c("P61247;PXXXXXX", "PYYYYYY;P46777;P60174", "PZZZZZZ"))
```

`CLES` *Common Language Effect Size*

Description

Calculate common language effect size.

Usage

```
CLES(x, y)
```

Arguments

`x`	numeric, data
`y`	numeric, data

Details

"For continuous data, it [the common language statistic] is 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)

This function calculates the fraction of all possible pairings between `x` and `y` where the difference ('`y`' - '`x`') is positive.

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. `http://www.cdc.gov/nchs/data/series/sr_11/sr11_238.pdf`

Examples

```
# generate synthetic data for heights in inches of 18-24
# year-old males and females (NCHS, 1987, Tables 13 and 14)
height_male <- rnorm(988, 69.7, 2.8)
height_female <- rnorm(1066, 64.3, 2.6)
# the CLES is approximately 0.92 (McGraw and Wong, 1992)
CLES(height_female, height_male)
```

`diffplot` *Plot Compositional Differences*

Description

Make a plot showing differences and *p*-values.

Usage

```
diffplot(comptab, vars = c("ZC", "nH2O"), col = "black",
         plot.rect = FALSE, pt.text = c(letters, LETTERS))
```

Arguments

`comptab`	list or data frame, summary of comparisons generated by `get_comptab`
`vars`	character, which variables to plot
`col`	the color(s) for points
`plot.rect`	plot a reference rectangle?
`pt.text`	text labels for the points

Details

A plot is created with points showing the differences in medians or means of two compositional metrics. The default setting of `vars` refers to average oxidation state of carbon (Z_{C}) as the x-variable and water demand per residue in formation reactions from basis species ($\bar{n}_{\mathrm{H_2O}}$) as the y-variable.

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.

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 `rect`angle is drawn with coordinates -0.01, -0.01, 0.01, 0.01. This is useful for emphasizing the different scales of adjacent plots.

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

Examples

```
library(CHNOSZ)
comptab <- lapply(c("JKMF10", "WDO+15_C.N"), function(dataset) {
  pdat <- get_pdat(dataset)
  get_comptab(pdat)
})
diffplot(comptab)
```

Ehplot *Plot Eh*

Description

Show redox potential (Eh) scale.

Usage

```
  Ehplot(basis = "QEC", T = 37, pH = 7.4)
```

Arguments

basis	character, keyword for basis species to use
T	numeric, temperature in degrees Celsius
pH	numeric, pH

Details

This function plots selected values of Eh (redox potential) as a function of $\log f_{\mathrm{O}_2}$ and $\log a_{\mathrm{H}_2\mathrm{O}}$. The lines are labeled with the Eh value in volts. The temperature and pH can be adjusted using the `T` and `pH` arguments (currently this only affects the lines, but not the positions of labels).

See Also

`convert` (from **CHNOSZ**)

Examples

```
library(CHNOSZ)
Ehplot()
```

`get_colors` *Get Colors*

Description

Get colors for rank-difference (potential) diagrams.

Usage

```
get_colors(x, max50 = FALSE)
```

Arguments

`x`	numeric values
`max50`	logical, use most intense color for all values >= 50?

Details

`get_colors` returns a diverging (blue - light gray - red) color scale. Blue and red colors are associated with negative and positive values, respectively. The intensity of the color increases with the magnitude of the value. For accurate representation, the values should be in a percent scale (i.e. the maximum absolute value is not greater than 100). By default, a value of +/- 100 corresponds to greatest intensity. Set `max50` to TRUE to compress the scale so that greatest intesity is obtained at values of +/- 50 and higher.

See Also

`rankplot`

`get_comptab` *Calculate Compositional Differences*

Description

Compare elemental abundances per residue and compositional oxidation and hydration state between groups of proteins.

Usage

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

Arguments

pdat	list, data object generated by get_pdat
var1	character, the first variable
var2	character, the second variable
plot.it	logical, make a scatterplot?
mfun	character, specifying use of the 'median' or 'mean' function

Details

get_comptab can be used to summarize differences of compositional variables between down (group 1)- and up (group 2)-expressed proteins. The available compositional variables are:

'ZC'	average oxidation state of carbon (Z_C; see ZC)
'nH2O'	water demand per residue ($\bar{n}_{H_2O}$)
'nC'	number of carbon atoms per residue
'nN'	number of nitrogen atoms per residue
'nS'	number of sulfur atoms per residue
'V0'	standard molal volume per residue
'nAA'	protein length (number of amino acids)

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

The expression pattern is taken from the value of up2 returned by one of the pdat_ functions: down (up2==FALSE) or up (up2==TRUE). The function prints the difference of medians (or means), common language effect size (CLES, in percent), and *p*-value.

If plot.it is TRUE, a scatterplot is also produced.

Value

A data frame is returned invisibly containing the columns 'dataset', 'description', 'n1', 'n2', and two sets of columns ('var.mfun1', 'var.mfun2', 'var.diff', 'var.CLES', 'var.p.value'), where 'var' is replaced by the values of var1 and var2, respectively, and 'mfun' is replaced by the value of mfun.

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. https://doi.org/10.5194/bg-3-311-2006

Examples

```
library(CHNOSZ)
pd <- get_pdat("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)
```

get_pdat *Get Protein Data*

Description

Get data on protein expression and chemical composition.

Usage

```
get_pdat(dataset = NULL, pdat_fun = "pdat_CRC", basis = "QEC")
```

Arguments

`dataset`	character, specifies which dataset to retrieve
`pdat_fun`	character, names of additional `pdat_` functions
`basis`	character, keyword for basis species to use

Details

`get_pdat` serves as a wrapper to the various `pdat_` functions. The names of the functions, which can include any user-defined functions in addition to those in the package, are specified in `pdat_fun`. Those functions are queried for the availability of `dataset`; if one function provides it, the corresponding dataset is returned. Leave `dataset` as `NULL` to list all datasets provided by the `pdat_` functions.

Examples

```
library(CHNOSZ)
# list available datasets from pdat_CRC()
get_pdat()
# the following produces the same result as pdat_CRC("JKMF10")
get_pdat("JKMF10")
```

`groupplots` *Plot Potential Diagrams for Groups of Datasets*

Description

Plot rank difference of chemical affinities for proteins in various datasets and merge the diagrams.

Usage

```
groupplots(group="hypoxia_ZC_down", each100=FALSE, res=50, plot.it=TRUE)
mergedplot(gpresult, each100=FALSE, res=50)
```

Arguments

`group`	character, description of datasets to include
`each100`	logical, rescale rank difference of each dataset individually?
`res`	numeric, grid resolution for plots
`plot.it`	logical, make plots?
`gpresult`	list, value returned by `groupplots`

Details

`groupplots` makes weighted rank-difference of affinity (potential) diagrams (see `rankplot`) for each dataset found in the specified `group`. `group` consists of three parts joined by an underscore: the type of experiment (‘`CRC`’, ‘`pancreatic`’, ‘`hypoxia`’, or ‘`osmotic`’; see `pdat_`), the distinguishing compositional variable (‘`ZC`’ or ‘`H2O`’), and the direction of change of that variable (‘`up`’ or ‘`down`’).

To identify the datasets in any group, compositional summaries for each dataset are read from pre-calculated tables in `extdata/summary` (see `pdat_`). Datasets are included for which the absolute mean difference of either ‘`ZC`’ or ‘`H2O`’ between up- and down-expressed proteins is greater than 0.01 and the other of ‘`ZC`’ or ‘`H2O`’ has *p*-value >= 0.05 and abs(`CLES` - 50) < 10.

`groupplots` makes calculations over a large range of $\log f_{O_2}$ and $\log a_{H_2O}$ in order to encompass the equipotential lines for most datasets. This way, the positions of the median and interquartiles of the equipotential lines can be calculated accurately for the `mergedplot`, which covers a smaller range of $\log f_{O_2}$ and $\log a_{H_2O}$.

See Also

`rankplot`

Examples

```
## Not run:
gpresult <- groupplots("osmotic_H2O_down", res=25)
mergedplot(gpresult, res=25)

# reproduce Figure 4 of Dick, 2017
```

```
ZCgroups <- c("CRC_ZC_up","pancreatic_ZC_up", "hypoxia_ZC_down")
H2Ogroups <- c("CRC_H2O_up", "pancreatic_H2O_up", "osmotic_H2O_down")
allgroups <- c(ZCgroups, H2Ogroups)
par(mfrow=c(2, 3))
for(group in allgroups) {
  gpresult <- groupplots(group, plot.it=FALSE)
  mergedplot(gpresult)
  title(main=group)
}
## End(Not run)
```

lapply_canprot *Parallel Computation*

Description

Set up and run parallel computations.

Usage

```
lapply_canprot(X, FUN, ..., varlist = NULL, min.length = 10)
```

Arguments

X	vector (atomic or list)
FUN	function or character string naming a function
...	additional arguments to pass to 'FUN'
varlist	character, names of variables to export
min.length	numeric, minimum length of X for running parallel computations

Details

If `length(X)` is less than `min.length`, this function simply calls `lapply` given the `X`, `FUN` and `...` arguments, and returns the result. Otherwise, a cluster is initiatied by loading **CHNOSZ** and **canprot** and setting up the respective data environments (`thermo` and `canprot`). If `varlist` is supplied, the named variables are exported to the global environment of each node in the cluster using `clusterExport`. Then, parallel computations in the cluster are run using `parLapply` given the `X`, `FUN` and `...` arguments.

For examples, see the data vignettes and the help for `pdat_`.

pdat_ *Get Protein Data*

Description

Get data on protein expression and chemical composition.

Usage

```
pdat_CRC(dataset = NULL, basis = "QEC")
```

Arguments

`dataset`	character, specifies which dataset to retrieve
`basis`	character, keyword for basis species to use

Details

The `pdat_` functions calculate chemical compositional metrics (using `protcomp`) for relatively up- and down-expressed proteins reported in proteomic experiments.

Use `pdat_CRC` to retrieve data for protein expression in colorectal cancer, `pdat_pancreactic` for data on pancreatic cancer, `pdat_hypoxia` for data on hypoxia or 3D culture, and `pdat_osmotic` for data on hyperosmotic stress. The functions get relative expression data from the CSV files stored in `extdata/expression/`, with subdirectories corresponding to the names of the functions. Some of the functions also retrieve amino acid compositions from the files in `extdata/aa/` (for non-human proteins).

If `dataset` is `NULL`, the return value gives the names of all datasets that can be retrieved using the function. Provide one of these names as the `dataset` argument to retrieve the data. Each dataset name indicates the study (publication) where the data were reported, constructed by combining the first characters of the (first three or four) authors' family names with the 2-digit year of publication. This coincides with the key-generation scheme used in some bibliography manager software. This abbreviation also is used to name the CSV file containing the data. If more than one dataset is available from a single study (for example, for relative protein expression in different stages of cancer), `dataset` is suffixed by an underscore followed by a short abbreviation indicating the particular dataset.

Tables listing mean compositional differences between up- and down-expressed proteins for each dataset are saved in `extdata/summary/`. These files were created using the second example below.

Value

A list consisting of `dataset` (the name of the dataset), `basis` (basis species used for the calculations), `description` (descriptive text), `pcomp` (compositional data generated by `protcomp`), `up2` (logical vector with length equal to the number of proteins; TRUE if the protein is up-expressed in group 2 compared to group 1 (i.e. cancer compared to normal), FALSE otherwise), `names` (gene names for the proteins, if available).

See Also

get_pdat

Examples

```
library(CHNOSZ)
pdat_CRC()
pdat_CRC("JKMF10")  # same result as get_pdat("JKMF10")

## Not run:
# how the extdata/summary/summary_*.csv files were made
for(what in c("CRC", "pancreatic", "hypoxia", "osmotic")) {
  pdat_fun <- paste0("pdat_", what)
  datasets <- get(pdat_fun)()
  comptab <- lapply_canprot(datasets, function(dataset) {
    pdat <- get_pdat(dataset, pdat_fun)
    get_comptab(pdat)
  }, varlist = "pdat_fun")
  # write summary table
  comptab <- do.call(rbind, comptab)
  comptab <- cbind(set = c(letters, LETTERS)[1:nrow(comptab)], comptab)
  comptab[, 6:15] <- signif(comptab[, 6:15], 4)
  filename <- paste0("summary_", what, ".csv")
  write.csv(comptab, filename, row.names = FALSE, quote = 3)
}
## End(Not run)
```

---

protcomp *Protein Compositions*

Description

Get amino acid and chemical compositions of proteins.

Usage

```
protcomp(uniprot = NULL, ip = NULL, basis = "QEC",
          aa_file = NULL, updates_file = NULL)
```

Arguments

uniprot	character, UniProt IDs of proteins
ip	numeric, indices of active proteins in CHNOSZ
basis	character, keyword for basis species to use
aa_file	character, file name
updates_file	character, file name

Details

This function retrieves the amino acid compositions of one or more proteins specified by `uniprot` or `ip`, then calculates some chemical compositional properties using functions provided by **CHNOSZ**. The `basis` argument is used to select the basis species using a keyword (see `basis`). For example, use ‘`CHNOS`’ for CO_2, NH_3, H_2S, H_2O, and O_2, or ‘`QEC`’ (the default) for glutamine, glutamic acid, cysteine, H_2O, and O_2.

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 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`). If `updates_file` is specified, these ID mappings are included alongside the packaged `uniprot_updates`.

Value

The function returns a list with elements `protein.formula` (elemental compositions of the proteins), `ZC` (average oxidation state of carbon), `protein.basis` (compositions of the proteins in terms of the basis species), `protein.length` (lengths of the amino acid sequences), `residue.basis` (per-residue compositions of the proteins in terms of the basis species), `residue.formula` (per-residue elemental compositions of the proteins), and `aa` (amino acid compositions of the proteins).

Examples

```
library(CHNOSZ)
protcomp("P24298")
```

`rankdiff` *Weighted Difference of Sums of Ranks*

Description

Calculate rank-sum difference between two groups, weighted by the sizes of the groups.

Usage

```
rankdiff(rank1, rank2, n1 = NULL, n2 = NULL, as.fraction=TRUE)
```

Arguments

`rank1`	numeric, ranks in group 1
`rank2`	numeric, ranks in group 2
`n1`	numeric, size of group 1
`n2`	numeric, size of group 2
`as.fraction`	logical, calculate the fraction of maximum possible difference?

Details

In a combined ranking of two groups, the comparison of sum of ranks has an easy interpretation only for groups of equal size. The weighted rank difference is used to compare groups of unequal size. The weighting ensures that 1) opposite extreme configurations give weighted rank differences with equal magnitudes, and 2) an evenly distributed (interspersed) ranking of the two groups has a weighted rank difference of zero (Dick, 2016).

If `n1` and `n2` are not given, `rank1` and `rank2` are interpreted as vectors holding the ranks for the two groups. If the sizes of the groups are supplied in `n1` and `n2`, then the single values or higher-dimensional objects in `rank1` and `rank2` are interpreted as the non-weighted sums of ranks of the two groups.

Examples

```
# rankings of H and C in H-H-H-H-C-C-C
rankdiff(1:4, 5:7, as.fraction=FALSE)  # 12
rankdiff(1:4, 5:7)  # 1

# rankings of H and C in C-C-C-H-H-H-H
rankdiff(4:7, 1:3, as.fraction=FALSE)  # -12
rankdiff(4:7, 1:3)  # -1

# rankings of H and C in H-C-H-C-H-C-H
rankdiff(c(1, 3, 5, 7), c(2, 4, 6))  # 0
```

`rankplot` *Plot Ranking of Chemical Affinities*

Description

Plot ranking of chemical affinities of groups of proteins.

Usage

```
  rankplot(pdat, T = 37, what = "rankdiff", main = NULL, res = 300,
           plot.it = TRUE, xlim = c(-75, -55), ylim = c(-10, 10))
```

Arguments

`pdat`	list, data object generated by `get_pdat`
`T`	numeric, temperature in degrees Celsius
`what`	character, “rankdiff” or “affinity”
`main`	character, text to use for title of plot
`res`	numeric, grid resolution for plot
`plot.it`	logical, draw a plot?
`xlim`	numeric, range of x-axis ($\log f_{O_2}$)
`ylim`	numeric, range of y-axis ($\log a_{H_2O}$)

Details

This function creates a $\log a_{H_2O}$ - $\log f_{O_2}$ diagram showing the relative stabilities of the two groups of proteins in the specified `dataset`. These groups consist of the relatively down- and up-expressed proteins identified by `up2` in one of the `pdat_` functions.

The function generates a colored `image` and `contour` plot showing the weighted difference of sums of ranks of chemical affinities of formation of proteins in the two groups (see `affinity` and `rankdiff`). Increasing intensity of blue or red colors represent higher rankings of the down-expressed (`up2==FALSE`) or up-expressed (`up2==TRUE`) proteins, respectively. Alternatively, if `what` is "affinity", a maximum affinity diagram is produced (see `diagram` in CHNOSZ), with fields colored red or blue according to the relative expression of the protein.

If `main` is NULL, the title for the plot is taken from the description supplied in `pdat`.

Set `plot.it` to FALSE to skip the plotting and instead return a list containing the computed rank differences and x- and y- values and labels.

See Also

`get_colors`

Examples

```
library(CHNOSZ)
pdat <- get_pdat("JKMF10")
rankplot(pdat, res=25)
rankplot(pdat, res=25, what="affinity")
```

`remove_entries` *Remove Data Entries*

Description

Remove rows of a data table matching a condition, and print an informative message.

Usage

```
remove_entries(dat, irm, dataset, description)
```

Arguments

`dat`	data frame
`irm`	logical, which rows to remove
`dataset`	character, name of dataset used in message
`description`	character, description used in message

Details

This function is used by the pdat_ functions to "clean up" particular datasets. Examples of the description in pdat_CRC are 'missing', 'conflicting', and 'duplicated'. This function is made accessible to the user who may wish to write their own pdat_ functions.

Examples

```
datadir <- paste0(system.file("extdata", package="canprot"), "/expression/CRC/")
dataset <- "STK+15"
# an excerpt from pdat_CRC() for dataset="STK+15"
dat <- read.csv(paste0(datadir, "STK+15.csv"), as.is=TRUE)
dat <- remove_entries(dat, is.na(dat$uniprot), dataset, "missing")
# the above prints "STK+15: dropping 3 missing proteins"
```

---

xsummary *Summarize Compositional Differences*

Description

Make an HTML table summarizing compositional differences.

Usage

```
xsummary(comptab, vars=c("ZC", "nH2O"))
```

Arguments

comptab	list or data frame, summary of comparisons generated by get_comptab
vars	character, two variables to tabulate

Details

This function makes an HTML table (using xtable) and adds bold and underline formatting to highlight significant compositional 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 median or mean 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).

Examples

```
library(CHNOSZ)
comptab <- lapply(c("JKMF10", "WDO+15_C.N"), function(dataset) {
  pdat <- get_pdat(dataset)
  get_comptab(pdat)
})
xsummary(comptab)
```

Index