Package 'wrProteo'

March 1, 2022

Version 1.6.0

Title Proteomics Data Analysis Functions

Author Wolfgang Raffelsberger [aut, cre]

Maintainer Wolfgang Raffelsberger <w.raffelsberger@gmail.com>

Description Data analysis of proteomics experiments by mass spectrometry is supported by this collection of functions mostly dedicated to the analysis of (bottom-up) quantitative (XIC) data.
Fasta-formatted proteomes (eg from UniProt Consortium <doi:10.1093/nar/gky1049>) can be read with automatic parsing and multiple annotation types (like species origin, abbreviated gene names, etc) extracted.
Quantitative proteomics measurements frequently contain multiple NA values, due to physical absence of given peptides in some samples, limitations in sensitivity or other reasons.
The functions provided here help to inspect graphically the data to investigate the nature of NA-values via their respective replicate measurements and to help/confirm the choice of NA-replacement by low random values.
Dedicated filtering and statistical testing using the framework of package 'limma' <doi:10.18129/B9.bioc.limma> can be run, enhanced by multiple rounds of NA-replacements to provide robustness towards rare stochastic events.
Multi-species samples, as frequently used in benchmark-tests (eg Navarro et al 2016 <doi:10.1038/nbt.3685>, Ramus et al 2016 <doi:10.1016/j.jprot.2015.11.011>), can be run with special options separating the data into sub-groups during normalization and testing.
As example the data-set from Ramus et al 2016 <doi:10.1016/j.jprot.2015.11.011>) is provided quantified by MaxQuant (Tyanova et al 2016 <doi:10.1038/nprot.2016.136>), ProteomeDiscoverer,
OpenMS (<doi:10.1038/nmeth.3959>) and Proline (Bouyssie et al 2020 <doi:10.1093/bioinformatics/btaa118>). Meta-data provided in sdrf format can be integrated to the analysis.
Subsequently, ROC curves (Hand and Till 2001 <doi:10.1023/A:1010920819831>) can be constructed to compare multiple analysis approaches.

Depends R (>= 3.5.0)

Imports grDevices, graphics, knitr, limma, stats, utils, wrMisc (>= 1.6.0)

Suggests data.table, fdrtool, MASS, RColorBrewer, readxl, ROTS, rmarkdown, R.utils, sm, wrGraph (>= 1.2.5)

License GPL-3

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2022-03-01 17:30:02 UTC

R topics documented:

AAmass *Molecular mass for amino-acids*

Description

Calculate molecular mass based on atomic composition

Usage

```
AAmass(massTy = "mono", inPept = TRUE, inclSpecAA = FALSE)
```

Arguments

massTy	(character) 'mono' or 'average'
inPept	(logical) remove H20 corresponding to water loss at peptide bond formaton
inclSpecAA	(logical) include ornithine O & selenocysteine U

Value

This function returns a vector with masses for all amino-acids (argument 'massTy' to switch form mono-isotopic to average mass)

See Also

massDeFormula, convToNum

Examples

```
massDeFormula(c("12H12O","HO"," 2H 1 Se, 6C 2N","HSeCN"," ","e"))
AAmass()
```

AucROC *AUC from ROC-curves*

Description

This function calculates the AUC (area under the curve) from ROC data in matrix of specificity and sensitivity values, as provided in the output from summarizeForROC.

Usage

```
AucROC(dat, useCol = c("spec", "sens"), silent = FALSE, callFrom = NULL)
```

Arguments

dat	(matrix or data.frame) main inut containig sensitivity and specificity data (from summarizeForROC)
useCol	(character or integer) column names to be used: 1st for specificity and 2nd for sensitivity count columns
silent	(logical) suppress messages
callFrom	(character) allows easier tracking of message(s) produced

Value

This functio returns a matrix including imputed values or list of final and matrix with number of imputed by group (plus optional plot)

See Also

preparing ROC data summarizeForROC, (re)plot the ROC figure plotROC; note that numerous other packages also provide support for working with ROC-curves : Eg rocPkgShort, ROCR, pROC or ROCit

Examples

```
set.seed(2019); test1 <- list(annot=cbind(spec=c(rep("b",35),letters[sample.int(n=3,
  size=150,replace=TRUE)])), BH=matrix(c(runif(35,0,0.01),runif(150)),ncol=1))
roc1 <- summarizeForROC(test1,spec=c("a","b","c"))
AucROC(roc1)
```

---

cleanListCoNames *Selective batch cleaning of sample- (ie column-) names in list*

Description

This function allows to manipulate sample-names (ie colnames) from data stored as multiple matrixes or data.frames in multiple sheets of a list in a batch-wise manner. Import functions such as readMaxQuantFile() organize initial flat files into lists (of matrixes) of the different types of data. Many times all column names in such lists carry long names including redundant information, like the overall experiment name or date, etc. The aim of this function is to facilitate 'cleaning' the sample- (ie column-) names to obtain short and concise names. Character terms to be removed (via argument rem) and/or replaced/subsitituted (via argument subst) should be given as they are, characters with special behaviour in grep (like '.') will be protected internally. Note, that the character substitution part will be done first, and the removal part (without character replacement) afterwards.

Usage

```
cleanListCoNames(
  dat,
  rem = NULL,
  subst = c("-", "_"),
  lstE = c("raw", "quant", "counts"),
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

dat	(list) main input
rem	(character) character string to be removed, may be named 'left' and 'right' for more specific exact pattern matching (this part will be perfomed before character substitutions by subst)
subst	(character of length=2, or matrix with 2 columns) pair(s) of character-strings for replacement (1st as search-item and 2nd as replacement); this part is performed after character-removal via rem
lstE	(character, length=1) names of list-elements where colnames should be cleaned
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of messages produced

Value

This function returns a list (equivalent to input dat)

See Also

grep

Examples

```
dat1 <- matrix(1:12, ncol=4, dimnames=list(1:3, paste0("sample_R.",1:4)))
dat1 <- list(raw=dat1, quant=dat1, notes="other..")
cleanListCoNames(dat1, rem=c(left="sample_"), c(".","-"))
```

---

combineMultFilterNAimput

Combine multiple filters on NA-imputed data

Description

In most omics data-analysis one needs to employ a certain number of filtering strategies to avoid getting artifacts to the step of statistical testing. `combineMultFilterNAimput` takes on one side the origial data and on the other side NA-imputed data to create several differnet filters and to finally combine them. A filter aiming to take away the least abundant values (using the imputede data) is fine-tuned by the argument `abundThr`. This step compares the means for each group and line, at least one grou-mean has to be > the threshold (based on hypothesis that if all conditions represent extrememy low measures their diffrenetial may not be determined with certainty). In contratst, the filter addressing the number of missing values (`NA`) uses the original data, the arguments `colTotNa`,`minSpeNo` and `minTotNo` are used at this step. Basically, this step allows defining a minimum content of 'real' (ie non-NA) values for further considering the measurements as reliable. This part uses internally `presenceFilt` for filtering elevated content of `NA` per line. Finally, this function combines both filters (as matrix of `FALSE` and `TRUE`) on NA-imputed and original data and retruns a vector of logical values if corresponding lines passe all filter criteria.

Usage

```
combineMultFilterNAimput(
  dat,
  imputed,
  grp,
  annDat = NULL,
  abundThr = NULL,
  colRazNa = NULL,
  colTotNa = NULL,
  minSpeNo = 1,
  minTotNo = 2,
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
```

Arguments

`dat`	(matrix or data.frame) main data (may contain `NA`)
`imputed`	(character) same as 'dat' but with all `NA` imputed
`grp`	(character or factor) define groups of replicates (in columns of 'dat')
`annDat`	(matrix or data.frame) annotation data (should match lines of 'dat')
`abundThr`	(numeric) optional threshold filter for minimumn abundance
`colRazNa`	(character) if razor peptides are used: column name for razor peptide count
`colTotNa`	(character) column name for total peptide count
`minSpeNo`	(integer) minimum number of specific peptides for maintaining proteins
`minTotNo`	(integer) minimum total ie max razor number of peptides
`silent`	(logical) suppress messages
`debug`	(logical) additional messages for debugging
`callFrom`	(character) allows easier tracking of messages produced

Value

This function returns a vector of logical values if corresponding line passes filter criteria

See Also

presenceFilt

Examples

```
set.seed(2013)
datT6 <- matrix(round(rnorm(300)+3,1), ncol=6,
  dimnames=list(paste0("li",1:50), letters[19:24]))
datT6 <- datT6 +matrix(rep(1:nrow(datT6),ncol(datT6)), ncol=ncol(datT6))
datT6[6:7,c(1,3,6)] <- NA
datT6[which(datT6 < 11 & datT6 > 10.5)] <- NA
datT6[which(datT6 < 6 & datT6 > 5)] <- NA
datT6[which(datT6 < 4.6 & datT6 > 4)] <- NA
datT6b <- matrixNAneighbourImpute(datT6, gr=gl(2,3))
datT6c <- combineMultFilterNAimput(datT6, datT6b, grp=gl(2,3), abundThr=2)
```

---

convAASeq2mass *Molecular mass for amino-acids*

Description

This function calculates the molecular mass of one-letter code amion-acid sequences.

Usage

```
convAASeq2mass(
  x,
  massTy = "mono",
  seqName = TRUE,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

x	(character) aminoacid sequence (single upper case letters for describing a peptide/protein)
massTy	(character) default 'mono' for mono-isotopic masses (alternative 'average')
seqName	(logical) optional (alternative) names for the content of 'x' (ie aa seq) as name (always if 'x' has no names)
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of message(s) produced

Value

This functions returns a vector with masses for all amino-acids (argument 'massTy' to switch form mono-isotopic to average mass)

See Also

massDeFormula, AAmass, convToNum

Examples

```
convAASeq2mass(c("PEPTIDE","fPROTEINES"))
pep1 <- c(aa="AAAA", de="DEFDEF")
convAASeq2mass(pep1, seqN=FALSE)
```

---

corColumnOrder *Order columns in list of matrixes (or matrix)*

Description

This function orders columns in list of matrixes (or matrix) according to argument sampNames. This function can be used to adjut/correct the order of samples after reading data using readMaxQuantFile(), readPDExport() etc. The input may also be MArrayLM-type object from package limma or from moderTestXgrp or moderTest2grp.

Usage

```
corColumnOrder(
  dat,
  sampNames,
  useListElem = c("quant", "raw"),
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

dat	(matrix, list or MArrayLM-object from limma) main input of which columns should get re-ordered, may be output from moderTestXgrp or moderTest2grp.
sampNames	(character) column-names in desired order for output
useListElem	(character) in case dat is list, all list-elements who's columns should get (re-)ordered
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of message(s) produced

Value

This function returns an object of same class as input `dat` (ie matrix, list or MArrayLM-object from limma)

See Also

`moderTestXgrp` for single comparisons, `order`

Examples

```
grp <- factor(rep(LETTERS[c(3,1,4)], c(2,3,3)))
dat1 <- matrix(1:15, ncol=5, dimnames=list(NULL,c("D","A","C","E","B")))
corColumnOrder(dat1, sampNames=LETTERS[1:5])

dat1 <- list(quant=dat1,raw=dat1)
  dat1
corColumnOrder(dat1, sampNames=LETTERS[1:5])
```

`countNoOfCommonPeptides`

Compare in-silico digested proteomes for unique and shared peptides, counts per protein or as peptides Compare in-silico digested proteomes for unique and shared peptides, counts per protein or as peptides. The in-silico digestion may be performed separately using the package R*hrefhttps://bioconductor.org/packages/release/bioc/html/cleaver.htmlcleaver. Note: input must be list (or multiple names lists) of proteins with their respective peptides (eg by in-silico digestion).*

Description

Compare in-silico digested proteomes for unique and shared peptides, counts per protein or as peptides

Compare in-silico digested proteomes for unique and shared peptides, counts per protein or as peptides. The in-silico digestion may be performed separately using the package cleaver. Note: input must be list (or multiple names lists) of proteins with their respective peptides (eg by in-silico digestion).

Usage

```
countNoOfCommonPeptides(
  ...,
  prefix = c("Hs", "Sc", "Ec"),
  sep = "_",
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

...	(list) multiple lists of (ini-silico) digested proteins (typically protein ID as names) with their respectice peptides (AA sequence), one entry for each species
prefix	(character) optional (species-) prefix for entries in '...', will be only considered if '...' has no names
sep	(character) concatenation symbol
silent	(logical) suppress messages
callFrom	(character) allows easier tracking of message(s) produced

Value

This function returns a list with $byPep as list of logical matrixes for each peptide (as line) and unique/shared/etc for each species; $byProt as list of matrixes with count data per proten (as line) for each species; $tab with simple summary-type count data

See Also

readFasta2 and/or cleave-methods in package cleaver

Examples

```
## The example mimics a proteomics experiment where extracts form E coli and
## Saccharomyces cerevisiae were mixed, thus not all peptdes may occur unique.
(mi2 = countNoOfCommonPeptides(Ec=list(E1=letters[1:4],E2=letters[c(3:7)],
  E3=letters[c(4,8,13)],E4=letters[9]),Sc=list(S1=letters[c(2:3,6)],
  S2=letters[10:13],S3=letters[c(5,6,11)],S4=letters[c(11)],S5="n")))
##  a .. uni E, b .. inteR, c .. inteR(+intra E), d .. intra E  (no4), e .. inteR,
##  f .. inteR +intra E   (no6), g .. uni E, h .. uni E  no 8), i .. uni E,
##  j .. uni S (no10), k .. intra S  (no11), l .. uni S (no12), m .. inteR  (no13)
lapply(mi2$byProt,head)
mi2$tab
```

---

extractTestingResults *Extract results from moderated t-tests*

Description

This function allows convenient access to results produced using the functions moderTest2grp or moderTestXgrp. The user can define the threshold which type of multiple testing correction should be used (as long as the multiple testing correction method was actually performed as part of testing).

Usage

```
extractTestingResults(
  stat,
  compNo = 1,
  statTy = "BH",
  thrsh = 0.05,
  FCthrs = 1.5,
  annotCol = c("Accession", "EntryName", "GeneName"),
  nSign = 6,
  addTy = c("allMeans"),
  filename = NULL,
  fileTy = "csvUS",
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

stat	('MArrayLM'-object or list) Designed for the output from moderTest2grp or moderTestXgrp
compNo	(integer) the comparison number/index to be used
statTy	(character) the multiple-testing correction type to be considered when looking for significant changes with threshold thrsh (depends on which have been run initially with moderTest2grp or moderTestXgrp)
thrsh	(numeric) the threshold to be applied on statTy for the result of the statistcal testing (after multiple testing correction)
FCthrs	(numeric) Fold-Change threshold given as Fold-change and NOT log2(FC), default at 1.5 (for filtering at M-value =0.585)
annotCol	(character) column-names from the annotation to be included
nSign	(integer) number of significant digits whe returning results
addTy	(character) additional groups to add (so far only "allMeans" available) in addition to the means used in the pairwise comparison
filename	(character) optional (path and) file-name for exporting results to csv-file
fileTy	(character) file-type to be used with argument filename, may be 'csvEur' or 'csvUS'
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of message(s) produced

Value

This function returns a limma-type MA-object (which can be handeled like a list)

See Also

moderTest2grp for single comparisons, moderTestXgrp for multiple comparisons, lmFit and the eBayes-family of functions in package limma

Examples

```
grp <- factor(rep(LETTERS[c(3,1,4)],c(2,3,3)))
set.seed(2017); t8 <- matrix(round(rnorm(208*8,10,0.4),2), ncol=8,
 dimnames=list(paste(letters[],rep(1:8,each=26),sep=""), paste(grp,c(1:2,1:3,1:3),sep="")))
t8[3:6,1:2] <- t8[3:6,1:2] +3                    # augment lines 3:6 (c-f)
t8[5:8,c(1:2,6:8)] <- t8[5:8,c(1:2,6:8)] -1.5    # lower lines
t8[6:7,3:5] <- t8[6:7,3:5] +2.2                  # augment lines
## expect to find C/A in c,d,g, (h)
## expect to find C/D in c,d,e,f
## expect to find A/D in f,g,(h)
library(wrMisc)     # for testing we'll use this package
test8 <- moderTestXgrp(t8, grp)
extractTestingResults(test8)
```

`extrSpeciesAnnot` *Extract species annotation*

Description

`extrSpeciesAnnot` identifies species-related annotation (as suffix to identifyers) for data comnining multiple species and returns alternative (short) names. This function also suppresses extra heading or tailing space or punctuation characters. In case multiple tags are found, the last tag is reported and a message of alert may be displayed.

Usage

```
extrSpeciesAnnot(
  annot,
  spec = c("_CONT", "_HUMAN", "_YEAST", "_ECOLI"),
  shortNa = c("cont", "H", "S", "E"),
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

`annot`	(character) vector with initial annotation
`spec`	(character) the tags to be identified
`shortNa`	(character) the final abbreviation used, order and lengt must fit to argument `annot`
`silent`	(logical) suppress messages
`callFrom`	(character) allows easier tracking of messages produced

Value

This function returns a character vector with single (last of multiple) term if found in argument `annot`

See Also

grep

Examples

```
spec <- c("keratin_CONT","AB_HUMAN","CD_YEAST","EF_G_HUMAN","HI_HUMAN_ECOLI","_YEAST_012")
extrSpeciesAnnot(spec)
```

---

foldChangeArrow2 *Add arrow for expected Fold-Change to VolcanoPlot or MA-plot*

Description

NOTE : This function is deprecated, please use foldChangeArrow instead !! This function was made for adding an arrow indicating a fold-change to MA- or Volcano-plots. When comparing mutiple concentratios of standards in benchmark-tests it may be useful to indicate the expected ratio in a pair-wise comparison. In case of main input as list or MArrayLM-object (as generated from limma), the colum-names of multiple pairwise comparisons can be used for extracting a numeric content (supposed as concentrations in sample-names) which will be used to determine the expected ratio used for plotting. Optionally the ratio used for plotting can be returned as numeric value.

Usage

```
foldChangeArrow2(
  FC,
  useComp = 1,
  isLin = TRUE,
  asX = TRUE,
  col = 2,
  arr = c(0.005, 0.15),
  lwd = NULL,
  addText = c(line = -0.9, cex = 0.7, txt = "expected", loc = "toright"),
  returnRatio = FALSE,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

FC	(numeric, list or MArrayLM-object) main information for drawing arrow : either numeric value for fold-change/log2-ratio of object to search for colnames of statistical testing for extracting numeric part
useComp	(integer) only used in case FC is list or MArrayLM-object an has multiple pairwise-comparisons
isLin	(logical) inidicate if FC is log2 or not
asX	(logical) indicate if arrow should be on x-axis

col	(integer or character) custom color
arr	(numeric, length=2) start- and end-points of arrow (as relative to entire plot)
lwd	(numeric) line-width of arrow
addText	(logical or named vector) indicate if text explaining arrow should be displayed, use TRUE for default (on top right of plot), or any combination of 'loc','line','cex','side','adj','col','text' (or 'txt') for customizing specific elements
returnRatio	(logical) return ratio
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of message(s) produced

Details

The argument addText also allows specifying a fixed position when using addText=c(loc="bottomleft"), also bottomright, topleft, topright, toleft and toright may be used. In this case the elemts side and adjust will be redefined to accomodate the text in the corner specified.

Ultimately this function will be integated to the package wrGraph.

Value

plots arrow only (and explicative text), if returnRatio=TRUE also returns numeric value for extracted ratio

See Also

new version : foldChangeArrow; used with MAplotW, VolcanoPlotW

Examples

```
plot(rnorm(20,1.5,0.1),1:20)
#deprecated# foldChangeArrow2(FC=1.5)
```

---

isolNAneighb *Isolate NA-neighbours*

Description

This functions extracts all replicate-values where at least one of the replicates is NA. Then, the non-NA values are sorted by the number of NAs which occored in this group of replicates. A list with all NA-neighbours organized by the number of NAs gets returned.

Usage

```
isolNAneighb(
  mat,
  gr,
  maxHi = 3,
  iniCheck = TRUE,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

mat	(matrix or data.frame) main data (may contain NA)
gr	(character or factor) grouping of columns of 'mat', replicate association
maxHi	(integer) maximum count of NAs to consider separately (higher ones will be counted/pooled as maxHi)
iniCheck	(logical) check at beginning if executing this function is useful (presence any NA)
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of messages produced

Value

This function returns a list with NA-neighbours sorted by number of NAs in replicate group

See Also

This function gets used by matrixNAneighbourImpute and testRobustToNAimputation; estimation of mode stableMode; detection of NAs na.fail

Examples

```
mat1 <- c(22.2, 22.5, 22.2, 22.2, 21.5, 22.0, 22.1, 21.7, 21.5, 22, 22.2, 22.7,
  NA, NA, NA, NA, NA, NA, NA, 21.2,   NA, NA, NA, NA,
  NA, 22.6, 23.2, 23.2,  22.4, 22.8, 22.8, NA,  23.3, 23.2, NA, 23.7,
  NA, 23.0, 23.1, 23.0,  23.2, 23.2, NA, 23.3,  NA, NA, 23.3, 23.8)
mat1 <- matrix(mat1, ncol=12, byrow=TRUE)
gr4 <- gl(3, 4)
isolNAneighb(mat1, gr4)
```

massDeFormula *molecular mass from chemical formula*

Description

Calculate molecular mass based on atomic composition

Usage

```
massDeFormula(
  comp,
  massTy = "mono",
  rmEmpty = FALSE,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

comp	(character) atomic compostion
massTy	(character) 'mono' or 'average'
rmEmpty	(logical) suppress empty entries
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of messages produced

Value

This function returns a numeric vector with mass

See Also

convToNum

Examples

```
massDeFormula(c("12H12O","HO"," 2H 1 Se, 6C 2N","HSeCN"," ","e"))
```

matrixNAinspect *Histogram of content of NAs in matrix*

Description

matrixNAinspect makes histograms of the full data and shows sub-population of NA-neighbour values. The aim of this function is to investigate the nature of NA values in matrix (of experimental measures) where replicate measurements are available. If a given element was measured twice, and one of these measurements revealed a NA while the other one gave a (finite) numeric value, the non-NA-value is considered a NA-neighbour. The subpopulation of these NA-neighbour values will then be highlighted in the resulting histogram. In a number of experimental settiongs some actual measurements may not meet an arbitrary defined baseline (as 'zero') or may be too low to be distinguishable from noise that associated measures were initially recorded as NA. In several types of measurments in proteomics and transcriptomics this may happen. So this fucntion allows to collect all NA-neighbour values and compare them to the global distribution of the data to investigate if NA-neighbours are typically very low values. In case of data with multiple replicates NA-neighbour values may be distinguished for the case of 2 NA per group/replicate-set. The resulting plots are typically used to decide if and how NA values may get replaced by imputed random values or wether measues containing NA-values should rather me omitted. Of course, such decisions do have a strong impact on further steps of data-analysis and should be performed with care.

Usage

```
matrixNAinspect(
  dat,
  gr,
  retnNA = TRUE,
  xLab = NULL,
  tit = NULL,
  xLim = NULL,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

dat	(matrix or data.frame) main numeric data
gr	(charcter or factor) grouping of columns of dat indicating who is a replicate of whom (ie the length of 'gr' must be equivalent to the number of columns in 'dat')
retnNA	(logical) report number of NAs in graphic
xLab	(character) custom x-label
tit	(character) custom title
xLim	(numerical,length=2) custom x-axis limits
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of messages produced

Value

This function produces graphics only

See Also

hist, na.fail, naOmit

Examples

```
set.seed(2013)
datT6 <- matrix(round(rnorm(300)+3,1),ncol=6,dimnames=list(paste("li",1:50,sep=""),letters[19:24]))
datT6 <- datT6 +matrix(rep(1:nrow(datT6),ncol(datT6)),ncol=ncol(datT6))
datT6[6:7,c(1,3,6)] <- NA
datT6[which(datT6 < 11 & datT6 > 10.5)] <- NA
datT6[which(datT6 < 6 & datT6 > 5)] <- NA
datT6[which(datT6 < 4.6 & datT6 > 4)] <- NA
matrixNAinspect(datT6,gr=gl(2,3))
```

---

matrixNAneighbourImpute

Imputation of NA-values based on non-NA replicates

Description

It is assumed that NA-values appear in data when quantitation values are very low (as this appears eg in quantitative shotgun proteomics). Here, the concept of (technical) replicates is used to investigate what kind of values appear in the other replicates next to NA-values for the same line/protein. Groups of replicate samples are defined via argument gr which descibes the columns of dat). Then, they are inspected for each line to gather NA-neighbour values (ie those values where NAs and regular measures are observed the same time). Eg, let's consider a line contains a set of 4 replicates for a given group. Now, if 2 of them are NA-values, the remaining 2 non-NA-values will be considered as NA-neighbours. Ultimately, the aim is to replaces all NA-values based on values from a normal distribution ressembling theire respective NA-neighbours.

Usage

```
matrixNAneighbourImpute(
  dat,
  gr,
  imputMethod = "mode2",
  retnNA = TRUE,
  avSdH = c(0.1, 0.5),
  NAneigLst = NULL,
  plotHist = c("hist", "mode"),
  xLab = NULL,
  xLim = NULL,
  yLab = NULL,
```

```
  yLim = NULL,
  tit = NULL,
  figImputDetail = TRUE,
  seedNo = NULL,
  silent = FALSE,
  callFrom = NULL,
  debug = FALSE
)
```

Arguments

dat	(matrix or data.frame) main data (may contain NA)
gr	(character or factor) grouping of columns of 'dat', replicate association
imputMethod	(character) choose the imputation method (may be 'mode2'(default), 'mode1', 'datQuant', 'modeAdopt' or 'informed')
retnNA	(logical) decide (if =TRUE) only NA-substuted data should be returned, or if list with $data, $nNA, $NAneighbour and $randParam should be returned
avSdH	(numerical,length=2) population characteristics 'high' (mean and sd) for >1 NA-neighbours (per line)
NAneigLst	(list) option for repeated rounds of imputations: list of NA-neighbour values can be furnished for slightly faster processing
plotHist	(character or logical) decide if supplemental figure with histogram shoud be drawn, the details 'Hist','quant' (display quantile of originak data), 'mode' (display mode of original data) can be chosen explicitely
xLab	(character) label on x-axis on plot
xLim	(numeric, length=2) custom x-axis limits
yLab	(character) label on y-axis on plot
yLim	(numeric, length=2) custom y-axis limits
tit	(character) title on plot
figImputDetail	(logical) display details about data (number of NAs) and imputation in graph (min number of NA-neighbours per protein and group, quantile to model, mean and sd of imputed)
seedNo	(integer) seed-value for normal random values
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of messages produced
debug	(logical) supplemental messages for debugging

Details

By default a histogram gets plotted showing the initial, imputed and final distribution to check the global hypothesis that NA-values arose from very low measurements and to appreciate the impact of the imputed values to the overall final distribution.

There are a number of experimental settings where low measurements may be reported as NA. Sometimes an arbitrary defined baseline (as 'zero') may provoke those values found below being unfortunately reported as NA or as 0 (in case of MaxQuant). In quantitative proteomics (DDA-mode) the

presence of numerous high-abundance peptides will lead to the fact that a number of less intense MS-peaks don't get identified properly and will then be reported as NA in the respective samples, while the same peptides may by correctly identified and quantified in other (replicate) samples. So, if a given protein/peptide gets properly quantified in some replicate samples but reported as NA in other replicate samples one may thus speculate that similar values like in the successful quantifications may have occored. Thus, imputation of NA-values may be done on the basis of NA-neighbours.

When extracting NA-neighbours, a slightly more focussed approach gets checked, too, the 2-NA-neighbours : In case a set of replicates for a given protein contains at least 2 non-NA-values (instead of just one) it will be considered as a (min) 2-NA-neighbour as well as regular NA-neighbour. If >300 of these (min) 2-NA-neighbours get found, they will be used instead of the regular NA-neighbours. For creating a collection of normal random values one may use directly the mode of the NA-neighbours (or 2-NA-neighbours, if >300 such values available). To do so, the first value of argument avSdH must be set to NA. Otherwise, the first value avSdH will be used as quantile of all data to define the mean for the imputed data (ie as `quantile(dat,avSdH[1],na.rm=TRUE)`). The sd for generating normal random values will be taken from the sd of all NA-neighbours (or 2-NA-neighbours) multiplied by the second value in argument avSdH (or avSdH, if >300 2-NA-neighbours), since the sd of the NA-neighbours is usually quite high. In extremely rare cases it may happen that no NA-neighbours are found (ie if NAs occur, all replicates are NA). Then, this function replaces NA-values based on the normal random values obtained as dscribed above.

Value

This function returns a list with `$data` .. matrix of data where NA are replaced by imputed values, `$nNA` .. number of NA by group, `$randParam` .. parameters used for making random data

See Also

this function gets used by `testRobustToNAimputation`; estimation of mode `stableMode`; detection of NAs `na.fail`

Examples

```
set.seed(2013)
datT6 <- matrix(round(rnorm(300)+3,1), ncol=6, dimnames=list(paste("li",1:50,sep=""),
  letters[19:24]))
datT6 <- datT6 +matrix(rep(1:nrow(datT6), ncol(datT6)), ncol=ncol(datT6))
datT6[6:7, c(1,3,6)] <- NA
datT6[which(datT6 < 11 & datT6 > 10.5)] <- NA
datT6[which(datT6 < 6 & datT6 > 5)] <- NA
datT6[which(datT6 < 4.6 & datT6 > 4)] <- NA
datT6b <- matrixNAneighbourImpute(datT6, gr=gl(2,3))
head(datT6b$data)
```

`plotROC` *Plot ROC curves*

Description

plotROC plots ROC curves based on results from summarizeForROC. This function plots only, it does not return any data. It allows printing simultaneously multiple ROC curves from different studies, it is also compatible with data from 3 species mix as in proteomics benchmark. Input can be prepared using moderTest2grp followed by summarizeForROC.

Usage

```
plotROC(
  dat,
  ...,
  useColumn = 2:3,
  methNames = NULL,
  col = NULL,
  pch = 1,
  bg = NULL,
  tit = NULL,
  xlim = NULL,
  ylim = NULL,
  point05 = 0.05,
  pointSi = 0.85,
  nByMeth = NULL,
  speciesOrder = NULL,
  txtLoc = NULL,
  legCex = 0.72,
  las = 1,
  addSuplT = TRUE,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

dat	(matrix) from testing (eg summarizeForROC)
...	optional additional data-sets to include as seprate ROC-curves to same plot (must be of same type of format as 'dat')
useColumn	(integer or character, length=2) columns from dat to be used for pecificity and sensitivity
methNames	(character) names of methods (data-sets) to be displayed
col	(character) custom colors for lines and text (choose one color for each different data-set)
pch	(integer) type of symbol to be used (see also par)
bg	(character) background color in plot (see also par)
tit	(character) custom title
xlim	(numeric, length=2) custom x-axis limits
ylim	(numeric, length=2) custom y-axis limits

point05	(numeric) specific point to highlight in plot (typically at alpha=0.05)
pointSi	(numeric) size of points (as expansion factor cex)
nByMeth	(integer) value of n to display
speciesOrder	(integer) custom order of species in legend
txtLoc	(numeric, length=3) location for text (x, y and proportional factor for line-offset, default is c(0.4,0.3,0.04))
legCex	(numeric) cex expansion factor for legend (see also par)
las	(numeric) factor for text-orientation (see also par)
addSuplT	(logical) add text with information about precision,accuracy and FDR
silent	(logical) suppress messages
callFrom	(character) allows easier tracking of messages produced

Value

This function returns only a plot with ROC curves

See Also

summarizeForROC, moderTest2grp

Examples

```
roc0 <- cbind(alph=c(2e-6,4e-5,4e-4,2.7e-3,1.6e-2,4.2e-2,8.3e-2,1.7e-1,2.7e-1,4.1e-1,5.3e-1,
 6.8e-1,8.3e-1,9.7e-1), spec=c(1,1,1,1,0.957,0.915,0.915,0.809,0.702,0.489,0.362,0.234,
  0.128,0.0426), sens=c(0,0,0.145,0.942,2.54,2.68,3.33,3.99,4.71,5.87,6.67,8.04,8.77,
  9.93)/10, n.pos.a=c(0,0,0,0,2,4,4,9,14,24,36,41) )
plotROC(roc0)
```

---

razorNoFilter *Filter based on either number of total peptides and specific peptides or number of razor petides*

Description

razorNoFilter filters based on either a) number of total peptides and specific peptides or b) numer of razor petides. This function was designed for filtering using a mimimum number of (PSM-) count values following the common practice to consider results with 2 or more peptide counts as reliable. The function be (re-)run independently on each of various questions (comparisons). Note: Non-integer data will be truncated to integer (equivalent to floor).

Usage

```
razorNoFilter(
  annot,
  speNa = NULL,
  totNa = NULL,
  minRazNa = NULL,
  minSpeNo = 1,
  minTotNo = 2,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

annot	(matrix or data.frame) main data (may contain NAs) with (PSM-) count values for each protein
speNa	(integer or character) indicate which column of 'annot' has number of specific peptides
totNa	(integer or character) indicate which column of 'annot' has number of total peptides
minRazNa	(integer or character) name of column with number of razor peptides, alternative to 'minSpeNo'& 'minTotNo'
minSpeNo	(integer) minimum number of pecific peptides
minTotNo	(integer) minimum total ie max razor number of peptides
silent	(logical) suppress messages
callFrom	(character) allows easier tracking of messages produced

Value

This function returns a vector of logical values if corresponding line passes filter criteria

See Also

presenceFilt

Examples

```
set.seed(2019); datT <- matrix(sample.int(20,60,replace=TRUE), ncol=6,
  dimnames=list(letters[1:10], LETTERS[1:6])) -3
datT[,2] <- datT[,2] +2
datT[which(datT <0)] <- 0
razorNoFilter(datT, speNa="A", totNa="B")
```

readFasta2 *Read file of protein sequences in fasta format Read fasta formatted file (from* R*hrefhttps://www.uniprot.orgUniProt) to extract (protein) sequences and name. If* tableOut=TRUE *output may be organized as matrix for separating meta-annotation (eg uniqueIdentifier, entryName, proteinName, GN) in separate columns.*

Description

Read file of protein sequences in fasta format

Read fasta formatted file (from UniProt) to extract (protein) sequences and name. If `tableOut=TRUE` output may be organized as matrix for separating meta-annotation (eg uniqueIdentifier, entryName, proteinName, GN) in separate columns.

Usage

```
readFasta2(
  filename,
  delim = "|",
  databaseSign = c("sp", "tr", "generic", "gi"),
  removeEntries = NULL,
  tableOut = FALSE,
  UniprSep = c("OS=", "OX=", "GN=", "PE=", "SV="),
  cleanCols = TRUE,
  silent = FALSE,
  callFrom = NULL,
  debug = FALSE
)
```

Arguments

`filename`	(character) names fasta-file to be read
`delim`	(character) delimeter at header-line
`databaseSign`	(character) characters at beginning right after the '>' (typically specifying the data-base-origin), they will be excluded from the sequance-header
`removeEntries`	(character) if `'empty'` allows removing entries without any sequence entries; set to `'duplicated'` to remove duplicate entries (same sequence and same header)
`tableOut`	(logical) toggle to return named character-vector or matrix with enhaced parsing of fasta-header. The resulting matrix will contain the comumns 'database','uniqueIdentifier','entryName', and further columns depending on argument `UniprSep`
`UniprSep`	(character) separators for further separating entry-fields if `tableOut=TRUE`, see also UniProt-FASTA-headers
`cleanCols`	(logical) remove columns with all entries NA, if `tableOut=TRUE`
`silent`	(logical) suppress messages
`callFrom`	(character) allows easier tracking of messages produced
`debug`	(logical) supplemental messages for debugging

Value

This function returns (depending on parameter `tableOut`) a) a simple character vector (of sequence) with Uniprot ID as name or b) a matrix with columns: 'database','uniqueIdentifier','entryName','proteinName','sequence' and further columns depending on argument `UniprSep`

See Also

`writeFasta2` for writing as fasta, or for reading `scan` or `read.fasta` from the package seqinr

Examples

```
# tiny example with common contaminants
path1 <- system.file('extdata',package='wrProteo')
fiNa <-  "conta1.fasta"
fasta1 <- readFasta2(file.path(path1,fiNa))
## now let's read and further separate annotation-fields
fasta2 <- readFasta2(file.path(path1,fiNa),tableOut=TRUE)
str(fasta1)
```

---

readMassChroQFile *Read tabulated files imported from MassChroQ*

Description

Quantification results using MassChroQ should be initially treated using the R-package MassChroqR (both distributed by the PAPPSO at http://pappso.inrae.fr/) for initial normalization on peptide-level and combination of peptide values into protein abundances.

Usage

```
readMassChroQFile(
  fileName,
  path = NULL,
  normalizeMeth = "median",
  sampleNames = NULL,
  refLi = NULL,
  separateAnnot = TRUE,
  tit = "MassChroQ",
  graphTit = NULL,
  wex = NULL,
  specPref = c(conta = "CON_|LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
  plotGraph = TRUE,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

fileName	(character) name of file to be read (default 'proteinGroups.txt' as typically generated by MaxQuant in txt folder)
path	(character) path of file to be read
normalizeMeth	(character) normalization method (will be sent to normalizeThis)
sampleNames	(character) new column-names for quantification data (ProteomeDiscoverer does not automatically use file-names from spectra)
refLi	(character or integer) custom specify which line of data is main species, if character (eg 'mainSpe'), the column 'SpecType' in $annot will be searched for exact match of the (single) term given
separateAnnot	(logical) if TRUE output will be organized as list with $annot, $abund for initial/raw abundance values and $quant with final normalized quantitations
tit	(character) custom title to plot
graphTit	(character) depreciated custom title to plot, please use 'tit'
wex	(integer) relative expansion factor of the violin-plot (will be passed to vioplotW)
specPref	(character or list) define characteristic text for recognizing (main) groups of species (1st for comtaminants - will be marked as 'conta', 2nd for main species-marked as 'mainSpe', and optional following ones for supplemental tags/species - maked as 'species2','species3',...); if list and list-element has multiple values they will be used for exact matching of accessions (ie 2nd of argument annotCol)
plotGraph	(logical) optional plot of type vioplot of initial and normalized data (using normalizeMeth); if integer, it will be passed to layout when plotting
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of messages produced

Details

The final output of this fucntion is a list containing 3 elements: $annot, $raw, $quant and $notes, or returns data.frame with entire content of file if separateAnnot=FALSE. Other list-elements remain empty to keep format compatible to other import functions.

This function has been developed using MassChroQ version 2.2 and R-package MassChroqR version 0.4.0. Both are distributed by the PAPPSO (http://pappso.inrae.fr/). When saving quantifications as RData, the ABUNDANCE_TABLE produced by mcq.get.compar(XICAB) should be used.

After import data get (re-)normalized according to normalizeMeth and refLi, and boxplots or vioplots drawn.

Value

This function returns list with $raw (initial/raw abundance values), $quant with final normalized quantitations, $annot, $counts an array with number of peptides, $quantNotes and $notes; or if separateAnnot=FALSE the function returns a data.frame with annotation and quantitation only

See Also

read.table, normalizeThis) , readProlineFile

Examples

```
path1 <- system.file("extdata", package="wrProteo")
fiNa <- "tinyMC.RData"
dataMC <- readMassChroQFile(file=fiNa, path=path1)
```

---

readMaxQuantFile *Read proteinGroups.txt files exported from MaxQuant*

Description

Quantification results form MaxQuant can be read using this function and relevant information extracted. Innput files compressed as .gz can be read as well. Besides protein abundance values (XIC) peptide counting information like number of unique razor-peptides or PSM values can be extracted, too. The protein abundance values mat be normalized using multiple methods (median normalization is default), the determination of normalization values can be restricted to specific proteins (normalization to bait protein(s), or to matrix in UPS1 spike-in experiments). Besides, a graphical display of the distruibution of protein abundance values may be generated.

Usage

```
readMaxQuantFile(
  path,
  fileName = "proteinGroups.txt",
  normalizeMeth = "median",
  quantCol = "LFQ.intensity",
  contamCol = "Potential.contaminant",
  pepCountCol = c("Razor + unique peptides", "Unique peptides", "MS.MS.count"),
  uniqPepPat = NULL,
  refLi = NULL,
 extrColNames = c("Majority.protein.IDs", "Fasta.headers", "Number.of.proteins"),
  specPref = c(conta = "conta|CON_|LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
  remRev = TRUE,
  separateAnnot = TRUE,
  tit = NULL,
  wex = 1.6,
  plotGraph = TRUE,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

path	(character) path of file to be read
fileName	(character) name of file to be read (default 'proteinGroups.txt' as typically generated by MaxQuant in txt folder). Gz-compressed files can be read, too.
normalizeMeth	(character) normalization method (for details see normalizeThis)
quantCol	(character or integer) exact col-names, or if length=1 content of quantCol will be used as pattern to search among column-names for $quant using grep
contamCol	(character or integer, length=1) which columns should be used for contaminants marked by ProteomeDiscoverer
pepCountCol	(character) pattern to search among column-names for count data (1st entry for 'Razor + unique peptides', 2nd fro 'Unique peptides', 3rd for 'MS.MS.count' (PSM))
uniqPepPat	(character, length=1) depreciated, please use pepCountCol instead
refLi	(character or integer) custom specify which line of data is main species, if character (eg 'mainSpe'), the column 'SpecType' in $annot will be searched for exact match of the (single) term given
extrColNames	(character) column names to be read (1: prefix for LFQ quantitation, default 'LFQ.intensity'; 2: column name for protein-IDs, default 'Majority.protein.IDs'; 3: column names of fasta-headers, default 'Fasta.headers', 4: column name for number of protein IDs matching, default 'Number.of.proteins')
specPref	(character) prefix to identifiers allowing to separate i) recognize contamination database, ii) species of main identifications and iii) spike-in species
remRev	(logical) option to remove all protein-identifications based on reverse-peptides
separateAnnot	(logical) if TRUE output will be organized as list with $annot, $abund for initial/raw abundance values and $quant with final normalized quantitations
tit	(character) custom title to plot
wex	(numeric) relative expansion factor of the violin in plot
plotGraph	(logical) optional plot vioplot of initial and normalized data (using normalizeMeth); alternatively the argument may contain numeric details that will be passed to layout when plotting
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of messages produced

Details

This function has been developed using MaxQuant versions 1.6.10.x to 2.0.x, the format of the resulting file 'proteinGroups.txt' is typically well conserved between versions. The final output is a list containing these elements: $raw, $quant, $annot, $counts, $quantNotes, $notes, or (if separateAnnot=FALSE) data.frame with annotation- and main quantification-content.

Value

This function returns a list with $raw (initial/raw abundance values), $quant with final normalized quantitations, $annot (columns), $counts an array with 'PSM' and 'NoOfRazorPeptides', $quantNotes and $notes; or a data.frame with quantitation and annotation if separateAnnot=FALSE

See Also

read.table, normalizeThis) , readProlineFile

Examples

```
path1 <- system.file("extdata", package="wrProteo")
# Here we'll load a short/trimmed example file (thus not the MaxQuant default name)
fiNa <- "proteinGroupsMaxQuant1.txt.gz"
specPr <- c(conta="conta|CON_|LYSC_CHICK", mainSpecies="YEAST",spike="HUMAN_UPS")
dataMQ <- readMaxQuantFile(path1, file=fiNa, specPref=specPr, tit="tiny MaxQuant")
summary(dataMQ$quant)
matrixNAinspect(dataMQ$quant, gr=gl(3,3))
```

---

readOpenMSFile *Read csv files exported by OpenMS*

Description

Protein quantification results form OpenMS which were exported as .csv can be imported and relevant information extracted. Peptide data get summarized by protein by top3 or sum methods. The final output is a list containing the elements: $annot, $raw, $quant ie normaized final quantifications, or returns data.frame with entire content of file if separateAnnot=FALSE.

Usage

```
readOpenMSFile(
  fileName = NULL,
  path = NULL,
  normalizeMeth = "median",
  refLi = NULL,
  sampleNames = NULL,
  quantCol = "Intensity",
  sumMeth = "top3",
  minPepNo = 1,
  protNaCol = "ProteinName",
  separateAnnot = TRUE,
  plotGraph = TRUE,
  tit = "OpenMS",
  wex = 1.6,
  specPref = c(conta = "LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
```

Arguments

fileName	(character) name of file to be read
path	(character) path of file to be read
normalizeMeth	(character) normalization method (will be sent to normalizeThis)
refLi	(character or integer) custom specify which line of data is main species, if character (eg 'mainSpe'), the column 'SpecType' in $annot will be searched for exact match of the (single) term given
sampleNames	(character) new column-names for quantification data (by default the names from files with spectra will be used)
quantCol	(character or integer) exact col-names, or if length=1 content of quantCol will be used as pattern to search among column-names for $quant using grep
sumMeth	(character) method for summarizing peptide data (so far 'top3' and 'sum' available)
minPepNo	(integer) minumun number of peptides to be used for retruning quantification
protNaCol	(character) column name to be read/extracted for the annotation section (default "ProteinName")
separateAnnot	(logical) if TRUE output will be organized as list with $annot, $abund for initial/raw abundance values and $quant with final normalized quantitations
plotGraph	(logical) optional plot of type vioplot of initial and normalized data (using normalizeMeth); if integer, it will be passed to layout when plotting
tit	(character) custom title to plot
wex	(integer) relative expansion factor of the violin-plot (will be passed to vioplotW)
specPref	(character or list) define characteristic text for recognizing (main) groups of species (1st for comtaminants - will be marked as 'conta', 2nd for main species-marked as 'mainSpe', and optional following ones for supplemental tags/species - maked as 'species2','species3',...); if list and list-element has multiple values they will be used for exact matching of accessions (ie 2nd of argument annotCol)
silent	(logical) suppress messages
debug	(logical) display additional messages for debugging
callFrom	(character) allow easier tracking of message(s) produced

Details

This function has been developed based on the OpenMS peptide-identification and label-free-quantification module. Csv input files may also be compresses as .gz.

Note: With this version the information about protein-modifications (PTMs) may not yet get exploited fully.

Value

This function returns a list with $raw (initial/raw abundance values), $quant with final normalized quantitations, $annot, $counts an array with number of peptides, $quantNotes,$expSetup and $notes; or if separateAnnot=FALSE the function returns a data.frame with annotation and quantitation only

See Also

read.table, normalizeThis) , readMaxQuantFile, readProlineFile, readProtDiscovFile

Examples

```
path1 <- system.file("extdata", package="wrProteo")
fiNa <- "OpenMS_tiny.csv.gz"
dataOM <- readOpenMSFile(file=fiNa, path=path1, tit="tiny OpenMS example")
summary(dataOM$quant)
```

---

readProlineFile *Read csv or txt files exported from Proline and MS-Angel*

Description

Quantification results form MS-Angel and Proline Proline exported as xlsx format can be read directly. Besides, files in tsv, csv (European and US format) or tabulated txt can be read, too. Then relevant information gets extracted, the data can optionally normalized and displayed as boxplot or vioplot. The final output is a list containing 6 elements: `$raw`, `$quant`, `$annot`, `$counts`, `$quantNotes` and `$notes`. Alternatively, a data.frame with annotation and quantitation data may be returned if `separateAnnot=FALSE`. Note: There is no normalization by default since quite frequently data produced by Proline are already sufficiently normalized. The figure produced using the argument `plotGraph=TRUE` may help judging if the data appear sufficiently normalized (distribtions should align).

Usage

```
readProlineFile(
  fileName,
  path = NULL,
  normalizeMeth = NULL,
  logConvert = TRUE,
  sampleNames = NULL,
  quantCol = "^abundance_",
  annotCol = c("accession", "description", "is_validated", "protein_set_score",
    "X.peptides", "X.specific_peptides"),
  remStrainNo = TRUE,
  pepCountCol = c("^psm_count_", "^peptides_count_"),
  trimColnames = FALSE,
  refLi = NULL,
  separateAnnot = TRUE,
  plotGraph = TRUE,
  tit = NULL,
  graphTit = NULL,
  wex = 2,
  specPref = c(conta = "_conta\\|", mainSpecies = "OS=Homo sapiens"),
```

```
  silent = FALSE,
  callFrom = NULL,
  debug = FALSE
)
```

Arguments

fileName	(character) name of file to read; .xlsx-, .csv-, .txt- and .tsv can be read (csv, txt and tsv may be gz-compressed). Reading xlsx requires package 'readxl'.
path	(character) optional path (note: Windows backslash sould be protected or written as '/')
normalizeMeth	(character) normalization method (for details and options see normalizeThis)
logConvert	(logical) convert numeric data as log2, will be placed in $quant
sampleNames	(character) new column-names for quantification data (ProteomeDiscoverer does not automatically use file-names from spectra); Please use with care since order of samples might be different as you expect
quantCol	(character or integer) colums with main quantitation-data : precise colnames to extract, or if length=1 content of quantCol will be used as pattern to search among column-names for $quant using grep
annotCol	(character) precise colnames or if length=1 pattern to search among column-names for $annot
remStrainNo	(logical) if TRUE, the organism annotation will be trimmed to uppercaseWord+space+lowercaseWord (eg Homo sapiens)
pepCountCol	(character) pattern to search among column-names for count data of PSM and NoOfPeptides
trimColnames	(logical) optional trimming of column-names of any redundant characters from beginning and end
refLi	(integer) custom decide which line of data is main species, if single character entry it will be used to choose a group of species (eg 'mainSpe')
separateAnnot	(logical) separate annotation form numeric data (quantCol and annotCol must be defined)
plotGraph	(logical or matrix of integer) optional plot vioplot of initial data; if integer, it will be passed to layout when plotting
tit	(character) custom title to plot
graphTit	(character) (depreciated custom title to plot), please use 'tit'
wex	(integer) relative expansion factor of the violin-plot (will be passed to vioplotW)
specPref	(character or list) define characteristic text for recognizing (main) groups of species (1st for comtaminants - will be marked as 'conta', 2nd for main species-marked as 'mainSpe', and optional following ones for supplemental tags/species - maked as 'species2','species3',...); if list and list-element has multiple values they will be used for exact matching of accessions (ie 2nd of argument annotCol)
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of message(s) produced
debug	(logical) display additional messages for debugging

Details

This function has been developed using Proline version 1.6.1 coupled with MS-Angel 1.6.1. The classical way of using ths function consists in exporting results produced by Proline and MS-Angel as xlsx file. Besides, other formats may be read, too. This includes csv (eg the main sheet/table of ths xlsx exported file saved as csv). WOMBAT represents an effort to automatize quantitative proteomics experiments, using this route data get exported as txt files which can be read, too.

Value

This function returns a list with `$raw` (initial/raw abundance values), `$quant` with final normalized quantitations, `$annot` (columns), `$counts` an array with 'PSM' and 'NoOfPeptides', `$quantNotes` and `$notes`; or a data.frame with quantitation and annotation if `separateAnnot=FALSE`

See Also

`read.table`

Examples

```
path1 <- system.file("extdata", package="wrProteo")
fiNa <- "exampleProlineABC.csv.gz"
dataABC <- readProlineFile(file.path(path1, fiNa))
summary(dataABC$quant)
```

---

readProtDiscovFile *Read tabulated files exported by ProteomeDiscoverer*

Description

Protein quantification results form Thermo ProteomeDiscoverer which were exported as tabulated text can be imported and relevant information extracted. The final output is a list containing 3 elements: `$annot`, `$raw` and optional `$quant`, or returns data.frame with entire content of file if `separateAnnot=FALSE`.

Usage

```
readProtDiscovFile(
  fileName,
  path = NULL,
  normalizeMeth = "median",
  sampleNames = NULL,
  infoFile = TRUE,
  read0asNA = TRUE,
  quantCol = "^Abundances*",
  annotCol = NULL,
  contamCol = "Contaminant",
  refLi = NULL,
```

```
  separateAnnot = TRUE,
 FDRCol = list(c("^Protein.FDR.Confidence", "High"), c("^Found.in.Sample.", "High")),
  plotGraph = TRUE,
  tit = "Proteome Discoverer",
  graphTit = NULL,
  wex = 1.6,
  specPref = c(conta = "CON_|LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
```

Arguments

fileName	(character) name of file to be read
path	(character) path of file to be read
normalizeMeth	(character) normalization method (will be sent to normalizeThis)
sampleNames	(character) new column-names for quantification data (ProteomeDiscoverer does not automatically use file-names from spectra); this argument has priority over infoFile
infoFile	(character or logical) filename containing additional information about MS-samples (produced by ProteomeDiscoverer default '*.InputFiles.txt'), if TRUE the first file in path containing the default name will be used. If no specific sampleNames given, the filenames will will be trimmed to remove redundant text and used as sampleNames. Besides, ProteomeDiscoverer version number and full raw-file path will be extracted for $notes in fial output.
read0asNA	(logical) decide if initial quntifications at 0 should be transformed to NA
quantCol	(character or integer) exact col-names, or if length=1 content of quantCol will be used as pattern to search among column-names for $quant using grep
annotCol	(character) column names to be read/extracted for the annotation section (default c("Accession","Description","Gene","Contaminant","Sum.PEP.Score","Coverage....","X..Peptides","X..P "X..AAs","MW..kDa."))
contamCol	(character or integer, length=1) which columns should be used for contaminants marked by ProteomeDiscoverer. If a column named contamCol is found, the data will be lateron filtered to remove all contaminants, set to NULL for keeping all contaminants
refLi	(character or integer) custom specify which line of data is main species, if character (eg 'mainSpe'), the column 'SpecType' in $annot will be searched for exact match of the (single) term given
separateAnnot	(logical) if TRUE output will be organized as list with $annot, $abund for initial/raw abundance values and $quant with final normalized quantitations
FDRCol	(list) optional indication to search for protein FDR information
plotGraph	(logical or integer) optional plot of type vioplot of initial and normalized data (using normalizeMeth); if integer, it will be passed to layout when plotting
tit	(character) custom title to plot

graphTit	(character) depreciated custom title to plot, please use 'tit'
wex	(integer) relative expansion factor of the violin-plot (will be passed to vioplotW)
specPref	(character or list) define characteristic text for recognizing (main) groups of species (1st for comtaminants - will be marked as 'conta', 2nd for main species-marked as 'mainSpe', and optional following ones for supplemental tags/species - maked as 'species2','species3',...); if list and list-element has multiple values they will be used for exact matching of accessions (ie 2nd of argument annotCol)
silent	(logical) suppress messages
debug	(logical) additional messages for debugging
callFrom	(character) allow easier tracking of messages produced

Details

This function has been developed using Thermo ProteomeDiscoverer versions 2.2 to 2.5. The format of resulting files at export also depends which columns are chosen as visible inside ProteomeDiscoverer and subsequently get chosen for export. Using the argument infoFile it is possible to specify a specific file (or search for default file) to read for extracting file-names as sample-names and other experiment realted information. If a column named contamCol is found, the data will be lateron filtered to remove all contaminants, set to NULL for keeping all contaminants This function replaces the depreciated function readPDExport.

Value

This function returns a list with $raw (initial/raw abundance values), $quant with final normalized quantitations, $annot, $counts an array with number of peptides, $quantNotes and $notes; or if separateAnnot=FALSE the function returns a data.frame with annotation and quantitation only

See Also

read.table, normalizeThis) , readMaxQuantFile, readProlineFile

Examples

```
path1 <- system.file("extdata", package="wrProteo")
fiNa <- "tinyPD_allProteins.txt.gz"
dataPD <- readProtDiscovFile(file=fiNa, path=path1)
summary(dataPD$quant)
```

readSdrf *Read proteomics meta-data as sdrf file This function allows reading proteomics meta-data from sdrf file, as they are provided on https://github.com/bigbio/proteomics-metadata-standard. Then, a data.frame with all annotation data will be returned. To stay conform with the (non-obligatory) recommendations, column-names will be shown as lower caps. The package utils must be installed.*

Description

Read proteomics meta-data as sdrf file

This function allows reading proteomics meta-data from sdrf file, as they are provided on https://github.com/bigbio/proteomic metadata-standard. Then, a data.frame with all annotation data will be returned. To stay conform with the (non-obligatory) recommendations, column-names will be shown as lower caps. The package utils must be installed.

Usage

```
readSdrf(
  fi,
  chCol = "auto",
  urlPrefix = "github",
  silent = FALSE,
  callFrom = NULL,
  debug = FALSE
)
```

Arguments

fi	(character) main input; may be full path or url to the file with meta-annotation. If a short project-name is given, it will be searched based at the location of urlPrefix
chCol	(character, length=1) optional checking of column-names
urlPrefix	(character, length=1) prefix to add to search when no complete path or url is given on fi, defaults to proteomics-metadata-standard on github
silent	(logical) suppress messages
callFrom	(character) allows easier tracking of message(s) produced
debug	(logical) display additional messages for debugging

Value

This function returns the content of Sdrf-file as data.frame

See Also

in read.table

Examples

```
pxd001819 <- readSdrf("PXD001819")
str(pxd001819)
```

`readUCSCtable` *Read annotation files from UCSC*

Description

This function allows reading and importing genomic UCSC-annotation data. Files can be read as default UCSC exprot or as GTF-format. In the context of proteomics we noticed that sometimes UniProt tables from UCSC are hard to match to identifiers from UniProt Fasta-files, ie many protein-identifiers won't match. For this reason additional support is given to reading 'Genes and Gene Predictions': Since this table does not include protein-identifiers, a non-redundant list of ENSxxx transcript identifiers can be exprted as file for an additional stop of conversion, eg using a batch conversion tool at the site of UniProt. The initial genomic annotation can then be complemented using `readUniProtExport`. Using this more elaborate route, we found higher coverage when trying to add genomic annotation to protein-identifiers to proteomics results with annnotation based on an initial Fasta-file.

Usage

```
readUCSCtable(
  fiName,
  exportFileNa = NULL,
  gtf = NA,
  simplifyCols = c("gene_id", "chr", "start", "end", "strand", "frame"),
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

`fiName`	(character) name (and path) of file to read
`exportFileNa`	(character) optional file-name to be exported, if `NULL` no file will be written
`gtf`	(logical) specify if file `fiName` in gtf-format (see UCSC)
`simplifyCols`	(character) optional list of column-names to be used for simplification (if 6 column-headers are given) : the 1st value will be used to identify the column used as refence to summarize all lines with this ID; for the 2nd (typically chromosome names) will be taken a representative value, for the 3rd (typically gene start site) will be taken the minimum, for the 4th (typically gene end site) will be taken the maximum, for the 5th and 6th a representative values will be reported;
`silent`	(logical) suppress messages
`callFrom`	(character) allows easier tracking of message(s) produced

Value

This function returns a matrix, optionally the file 'exportFileNa' may be written

See Also

readUniProtExport

Examples

```
path1 <- system.file("extdata", package="wrProteo")
gtfFi <- file.path(path1, "UCSC_hg38_chr11extr.gtf.gz")
# here we'll write the file for UniProt conversion to tempdir() to keep things tidy
expFi <- file.path(tempdir(), "deUcscForUniProt2.txt")
UcscAnnot1 <- readUCSCtable(gtfFi, exportFileNa=expFi)

## results can be further combined with readUniProtExport()
deUniProtFi <- file.path(path1, "deUniProt_hg38chr11extr.tab")
deUniPr1 <- readUniProtExport(deUniProtFi, deUcsc=UcscAnnot1,
  targRegion="chr11:1-135,086,622")
deUniPr1[1:5,-5]
```

---

readUniProtExport *Read protein annotation as exported from UniProt batch-conversion*

Description

This function allows reading and importing protein-ID conversion results from UniProt. To do so, first copy/paste your query IDs into UniProt 'Retrieve/ID mapping' field called '1. Provide your identifiers' (or upload as file), verify '2. Select options'. In a typical case of 'enst000xxx' IDs you may leave default settings, ie 'Ensemble Transcript' as input and 'UniProt KB' as output. Then, 'Submit' your search and retreive results via 'Download', you need to specify a 'Tab-separated' format ! If you download as 'Compressed' you need to decompress the .gz file before running the function `readUCSCtable` In addition, a file with UCSC annotation (Ensrnot accessions and chromosomic locations, obtained using `readUCSCtable`) can be integrated.

Usage

```
readUniProtExport(
  UniProtFileNa,
  deUcsc = NULL,
  targRegion = NULL,
  useUniPrCol = NULL,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

UniProtFileNa	(character) name (and path) of file exported from Uniprot (tabulated text file inlcuding headers)
deUcsc	(data.frame) object produced by readUCSCtable to be combined with data from UniProtFileNa
targRegion	(character or list) optional marking of chromosomal locations to be part of a given chromosomal target region, may be given as character like chr11:1-135,086,622 or as list with a first component characterizing the chromosome and a integer-vector with start- and end- sites
useUniPrCol	(character) optional declaration which colums from UniProt exported file should be used/imported (default 'EnsID','Entry','Entry.name','Status','Protein.names','Gene.names','Length').
silent	(logical) suppress messages
callFrom	(character) allows easier tracking of message(s) produced

Details

In a typicall use case, first chromosomic location annotation is extracted from UCSC for the species of interest and imported to R using readUCSCtable . However, the tables provided by UCSC don't contain Uniprot IDs. Thus, an additional (batch-)conversion step needs to get added. For this reason readUCSCtable allows writing a file with Ensemble transcript IDs which can be converted tu UniProt IDs at the site of UniProt. Then, UniProt annotation (downloaded as tab-separated) can be imported and combined with the genomic annotation using this function.

Value

This function returns a data.frame (with columns $EnsID, $Entry, $Entry.name, $Status, $Protein.names, $Gene.names, $Length; if deUcsc is integrated plus: $chr, $type, $start, $end, $score, $strand, $Ensrnot, $avPos)

See Also

readUCSCtable

Examples

```
path1 <- system.file("extdata",package="wrProteo")
deUniProtFi <- file.path(path1,"deUniProt_hg38chr11extr.tab")
deUniPr1a <- readUniProtExport(deUniProtFi)
str(deUniPr1a)

## Workflow starting with UCSC annotation (gtf) files :
gtfFi <- file.path(path1,"UCSC_hg38_chr11extr.gtf.gz")
UcscAnnot1 <- readUCSCtable(gtfFi)
## Results of conversion at UniProt are already available (file "deUniProt_hg38chr11extr.tab")
myTargRegion <- list("chr1", pos=c(198110001,198570000))
myTargRegion2 <-"chr11:1-135,086,622"      # works equally well
deUniPr1 <- readUniProtExport(deUniProtFi,deUcsc=UcscAnnot1,
  targRegion=myTargRegion)
```

```
## Now UniProt IDs and genomic locations are both available :
str(deUniPr1)
```

`removeSampleInList` *Remove samples/columns from list of matrixes Remove samples (ie columns) from every instance of list of matrixes. Note: This function assumes same order of columns in list-elements 'listElem' !*

Description

Remove samples/columns from list of matrixes

Remove samples (ie columns) from every instance of list of matrixes. Note: This function assumes same order of columns in list-elements 'listElem' !

Usage

```
removeSampleInList(
  dat,
  remSamp,
  listElem = c("abund", "quant"),
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

`dat`	(list) main input to be filtered
`remSamp`	(integer) column number to exclude
`listElem`	(character) names of list-elements where columns indicated with 'remSamp' should be removed
`silent`	(logical) suppress messages
`callFrom`	(character) allows easier tracking of messages produced

Value

This function returns a matrix including imputed values or list of final and matrix with number of imputed by group (plus optional plot)

See Also

`testRobustToNAimputation`

Examples

```
set.seed(2019)
datT6 <- matrix(round(rnorm(300)+3,1), ncol=6, dimnames=list(paste("li",1:50,sep=""),
  letters[19:24]))
datL <- list(abund=datT6, quant=datT6, annot=matrix(nrow=nrow(datT6), ncol=2))
datDelta2 <- removeSampleInList(datL, remSam=2)
```

`replMissingProtNames` *Complement missing EntryNames in annotation*

Description

This function helps replacing missing EntryNames (in $annot) after reading quantification results. To do so the comumn-names of `annCol` will be used : The content of 2nd element (and optional 3rd element) will be used to replace missing content in column defined by 1st element.

Usage

```
replMissingProtNames(
  x,
  annCol = c("EntryName", "Accession", "SpecType"),
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

`x`	(list) output of `readMaxQuantFile`, `readProtDiscovFile` or `readProlineFile`. This list must be a matrix and contain $annot with the columns designated in `annCol`.
`annCol`	(character) the column-names form `x$annot`) which will be used : The first column designs the column where empty fields are searched and the 2nd and (optional) 3rd will be used to fill the empty spots in the st column
`silent`	(logical) suppress messages
`callFrom`	(character) allow easier tracking of message(s) produced

Value

This function returns a list (like as input), but with missing elments of $annot completed (if available in other columns)

See Also

`readMaxQuantFile`, `readProtDiscovFile`, `readProlineFile`

Examples

```
dat <- list(quant=matrix(sample(11:99,9,replace=TRUE), ncol=3), annot=cbind(EntryName=c(
 "YP010_YEAST","",""),Accession=c("A5Z2X5","P01966","P35900"), SpecType=c("Yeast",NA,NA)))
replMissingProtNames(dat)
```

---

summarizeForROC *Summarize statistical test result for plotting ROC-curves*

Description

summarizeForROC takes statistical testing results (obtained using testRobustToNAimputation or moderTest2grp, based on limma) and calculates specifcity and sensitivity values for plotting ROC-curves along a panel of thresholds. Based on column from test$annot and argument 'spec' TP,FP,FN and TN are determined. Special consideration is made to 3 species mix samples as found in proteomics benchmark-tests. See also ROC on Wkipedia for explanations of TP,FP,FN and TN as well as examples. An optional plot may be produced, too. Return matrix with TP,FP,FN,TN,spec,sens,prec,accur and FDR count values along the various thrsholds specified in column 'alph'. Note that numerous other packages also provide support for building and plotting ROC-curves : Eg rocPkgShort, ROCR, pROC or ROCit

Usage

```
summarizeForROC(
  test,
  useComp = 1,
  tyThr = "BH",
  thr = NULL,
  columnTest = NULL,
  FCthrs = NULL,
  spec = c("H", "E", "S"),
  annotCol = "spec",
  filterMat = "filter",
  tit = NULL,
  color = 1,
  plotROC = TRUE,
  pch = 1,
  bg = NULL,
  overlPlot = FALSE,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

test	(class MArrayLM, S3-object from limma) from testing (eg testRobustToNAimputation or test2grp

useComp	(character or integer) in case multiple comparisons (ie multiple columns 'test$tyThr'); which pairwise comparison to used
tyThr	(character,length=1) type of stat test-result to be used for sensitivity and specificity calculations (eg 'BH','lfdr' or 'p.value'), must be list-element of 'test'
thr	(numeric) stat test (FDR/p-value) threshold, if NULL a panel of 108 p-value threshold-levels values will be used for calculating specifcity and sensitivity
columnTest	depreciated, please use 'useComp' instead
FCthrs	(numeric) Fold-Change threshold (display as line) give as Fold-change and NOT log2(FC), default at 1.5, set to NA for omitting
spec	(character) labels for species will be matched to column 'spec' of test$annot and used for sensitivity and specificity calculations. Important : 1st label for matrix (expected as constant) and subsequent labels for spike-ins (variable)
annotCol	(character) column name of test$annot to use to separate species
filterMat	(character) name (or index) of element of test containing matrix or vector of logical filtering results
tit	(character) optinal custom title in graph
color	(character or integer) color in graph
plotROC	(logical) toogle plot on or off
pch	(integer) type of symbol to be used (see par)
bg	(character) backgroud in plot (see par)
overlPlot	(logical) overlay to existing plot if TRUE
silent	(logical) suppress messages
callFrom	(character) allows easier tracking of message(s) produced

Value

This function returns a matrix including imputed values or list of final and matrix with number of imputed by group (plus optional plot)

See Also

replot the figure plotROC, calculate AUC using AucROC, robust test for preparing tables testRobustToNAimputation, moderTest2grp, test2grp, eBayes in package limma, t.test

Examples

```
set.seed(2019); test1 <- list(annot=cbind(spec=c(rep("b",35), letters[sample.int(n=3,
  size=150, replace=TRUE)])), BH=matrix(c(runif(35,0,0.01), runif(150)), ncol=1))
tail(roc1 <- summarizeForROC(test1, spec=c("a","b","c")))
```

test2grp *t-test each line of 2 groups of data*

Description

test2grp performs t-test on two groups of data using limma, this is a custom implementation of moderTest2grp for proteomics. The final obkect also includes the results without moderation by limma (eg BH-FDR in $nonMod.BH). Furthermore, there is an option to make use of package ROTS (note, this will increase the time of computatins considerably).

Usage

```
test2grp(
  dat,
  questNo,
  useCol = NULL,
  grp = NULL,
  annot = NULL,
  ROTSn = 0,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

dat	(matrix or data.frame) main data (may contain NAs)
questNo	(integer) specify here which question, ie comparison should be adressed
useCol	(integer or character)
grp	(character or factor)
annot	(matrix or data.frame)
ROTSn	(integer) number of iterations ROTS runs (stabilization of reseults may be seen with >300)
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of messages produced

Value

This function returns a limma-type S3 object of class 'MArrayLM' (which can be accessed like a list); multiple testing correction types or modified testing by ROTS may get included ('p.value','FDR','BY','lfdr' or 'ROTS.BH')

See Also

moderTest2grp, pVal2lfdr, t.test, ROTS from the Bioconductor package ROTS

Examples

```
set.seed(2018);  datT8 <- matrix(round(rnorm(800)+3,1), nc=8, dimnames=list(paste(
  "li",1:100,sep=""), paste(rep(LETTERS[1:3],c(3,3,2)),letters[18:25],sep="")))
datT8[3:6,1:2] <- datT8[3:6,1:2] +3   # augment lines 3:6 (c-f)
datT8[5:8,5:6] <- datT8[5:8,5:6] +3   # augment lines 5:8 (e-h)
grp8 <- gl(3,3,labels=LETTERS[1:3],length=8)
datL <- list(data=datT8, filt= wrMisc::presenceFilt(datT8,grp=grp8,maxGrpM=1,ratMa=0.8))
testAvB0 <- wrMisc::moderTest2grp(datT8[,1:6], gl(2,3))
testAvB <- test2grp(datL, questNo=1)
```

---

testRobustToNAimputation

Pair-wise testing robust to NA-imputation

Description

testRobustToNAimputation replaces NA values based on group neighbours (based on grouping of columns in argument gr), following overall assumption of close to Gaussian distribution. Furthermore, it is assumed that NA-values originate from experimental settings where measurements at or below detection limit are recoreded as NA. In such cases (eg in proteomics) it is current practice to replace NA-values by very low (random) values in order to be able to perform t-tests. However, random normal values used for replacing may in rare cases deviate from the average (the 'assumed' value) and in particular, if multiple NA replacements are above the average, may look like induced biological data and be misinterpreted as so. The statistical testing uses eBayes from Bioconductor package limma for robust testing in the context of small numbers of replicates. By repeating multiple times the process of replacing NA-values and subsequent testing the results can be sumarized afterwards by median over all repeated runs to remmove the stochastic effect of individual NA-imputation. Thus, one may gain stability towards random-character of NA imputations by repeating imputation & test 'nLoop' times and summarize p-values by median (results stabilized at 50-100 rounds). It is necessary to define all groups of replicates in gr to obtain all possible pair-wise testing (multiple columns in $BH, $lfdr etc). The modified testing-procedure of Bioconductor package ROTS may optionaly be included, if desired. This function returns a limma-like S3 list-object further enriched by additional fields/elements.

Usage

```
testRobustToNAimputation(
  dat,
  gr,
  annot = NULL,
  retnNA = TRUE,
  avSdH = c(0.15, 0.5),
  avSdL = NULL,
  plotHist = FALSE,
  xLab = NULL,
  tit = NULL,
  imputMethod = "mode2",
```

```
  seedNo = NULL,
  multCorMeth = NULL,
  nLoop = 100,
  lfdrInclude = NULL,
  ROTSn = NULL,
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
```

Arguments

dat	(matrix or data.frame) main data (may contain NA); if dat is list containing $quant and $annot as matrix, the element $quant will be used
gr	(character or factor) replicate association
annot	(matrix or data.frame) annotation (lines must match lines of data !), if annot is NULL and argument dat is a list containing both $quant and $annot, the element $annot will be used
retnNA	(logical) retain and report number of NA
avSdH	(numeric) population characteristics (mean and sd) for >1 NA neighbours 'high' (per line)
avSdL	depreciated argument, no longer used
plotHist	(logical) additional histogram of original, imputed and resultant distribution (made using matrixNAneighbourImpute)
xLab	(character) custom x-axis label
tit	(character) custom title
imputMethod	(character) choose the imputation method (may be 'mode2'(default), 'mode1', 'datQuant', 'modeAdopt' or 'informed', for details see matrixNAneighbourImpute)
seedNo	(integer) seed-value for normal random values
multCorMeth	(character) define which method(s) for correction of multipl testing should be run (for choice : 'BH','lfdr','BY','tValTab', choosing several is possible)
nLoop	(integer) number of runs of independent NA-imputation
lfdrInclude	(logical) depreciated, please used multCorMeth instead (include lfdr estimations, may cause warning message(s) concerning convergence if few too lines/proteins in dataset tested).
ROTSn	(integer) depreciated, please used multCorMeth instead (number of repeats by ROTS, if NULL ROTS will not be called)
silent	(logical) suppress messages
debug	(logical) additional messages fro debugging
callFrom	(character) allows easier tracking of messages produced

Details

The argument `multCorMeth` allows to choose which multiple correction algorimths will be used and included to the final results. Possible options are 'lfdr','BH','BY','tValTab', ROTSn='100' (name to element necessary) or 'noLimma' (to add initial p.values and BH to limma-results). By default 'lfdr' (local false discovery rate from package 'fdrtools') and 'BH' (Benjamini-Hochberg FDR) are chosen. The option 'BY' referrs to Benjamini-Yakuteli FDR, 'tValTab' allows exporting all individual t-values from the repeated NA-substitution and subsequent testing.

Value

This function returns a limma-type S3 object of class 'MArrayLM' (which can be accessed lika a list); multiple results of testing or multiple testing correction types may get included ('p.value','FDR','BY','lfdr' or 'ROTS.BH')

See Also

`moderTest2grp`, `pVal2lfdr`, `eBayes` in Bioconductor package limma, `t.test`,`ROTS` of Bioconductor package ROTS

Examples

```
set.seed(2015); rand1 <- round(runif(600) +rnorm(600,1,2),3)
dat1 <- matrix(rand1,ncol=6) + matrix(rep((1:100)/20,6),ncol=6)
dat1[13:16,1:3] <- dat1[13:16,1:3] +2      # augment lines 13:16
dat1[19:20,1:3] <- dat1[19:20,1:3] +3      # augment lines 19:20
dat1[15:18,4:6] <- dat1[15:18,4:6] +1.4    # augment lines 15:18
dat1[dat1 <1] <- NA                        # mimick some NAs for low abundance
## normalize data
boxplot(dat1, main="data before normalization")
dat1 <- wrMisc::normalizeThis(as.matrix(dat1), meth="median")
## designate replicate relationships in samples ...
grp1 <- gl(2, 3, labels=LETTERS[1:2])
## moderated t-test with repeated inputations (may take >10 sec,  >60 sec if ROTSn >0 !)
PLtestR1 <- testRobustToNAimputation(dat=dat1, gr=grp1, retnNA=TRUE, nLoop=70)
names(PLtestR1)
```

VolcanoPlotW2 *Volcano-plot (Statistical Test Outcome versus Relative Change)*

Description

This type of plot is very common in high-throughput biology, see Volcano-plot. Basically, this plot allows comparing the outcome of a statistical test to the differential of the group means (ie log fold-change),

Usage

```
VolcanoPlotW2(
  Mvalue,
  pValue = NULL,
  useComp = 1,
  filtFin = NULL,
  ProjNa = NULL,
  FCthrs = NULL,
  FdrList = NULL,
  FdrThrs = NULL,
  FdrType = NULL,
  subTxt = NULL,
  grayIncrem = TRUE,
  col = NULL,
  pch = 16,
  compNa = NULL,
  batchFig = FALSE,
  cexMa = 1.8,
  cexLa = 1.1,
  limM = NULL,
  limp = NULL,
  annotColumn = c("SpecType", "GeneName", "EntryName", "Accession", "Species",
    "Contam"),
  annColor = NULL,
  cexPt = NULL,
  cexSub = NULL,
  cexTxLab = 0.7,
  namesNBest = NULL,
  NbestCol = 1,
  sortLeg = "descend",
  NaSpecTypeAsContam = TRUE,
  useMar = c(6.2, 4, 4, 2),
  returnData = FALSE,
  callFrom = NULL,
  silent = FALSE,
  debug = FALSE
)
```

Arguments

Mvalue	(numeric or matrix) data to plot; M-values are typically calculated as difference of log2-abundance values and 'pValue' the mean of log2-abundance values; M-values and p-values may be given as 2 columsn of a matrix, in this case the argument pValue should remain NULL
pValue	(numeric, list or data.frame) if NULL it is assumed that 2nd column of 'Mvalue' contains the p-values to be used
useComp	(integer, length=1) choice of which of multiple comparisons to present in Mvalue (if generated using moderTestXgrp())

filtFin	(matrix or logical) The data may get filtered before plotting: If FALSE no filtering will get applied; if matrix of TRUE/FALSE it will be used as optional custom filter, otherwise (if Mvalue if an MArrayLM-object eg from limma) a default filtering based on the filtFin element will be applied
ProjNa	(character) custom title
FCthrs	(numeric) Fold-Change threshold (display as line) give as Fold-change and NOT log2(FC), default at 1.5, set to NA for omitting
FdrList	(numeric) FDR data or name of list-element
FdrThrs	(numeric) FDR threshold (display as line), default at 0.05, set to NA for omitting
FdrType	(character) FDR-type to extract if Mvalue is 'MArrayLM'-object (eg produced by from moderTest2grp etc); if NULL it will search for suitable fields/values in this order : 'FDR','BH',"lfdr" and 'BY'
subTxt	(character) custom sub-title
grayIncrem	(logical) if TRUE, display overlay of points as increased shades of gray
col	(character) custom color(s) for points of plot (see also par)
pch	(integer) type of symbol(s) to plot (default=16) (see also par)
compNa	(character) names of groups compared
batchFig	(logical) if TRUE figure title and axes legends will be kept shorter for display on fewer splace
cexMa	(numeric) font-size of title, as expansion factor (see also cex in par)
cexLa	(numeric) size of axis-labels, as expansion factor (see also cex in par)
limM	(numeric, length=2) range of axis M-values
limp	(numeric, length=2) range of axis FDR / p-values
annotColumn	(character) column names of annotation to be extracted (only if Mvalue is MArrayLM-object containing matrix $annot). The first entry (typically 'SpecType') is used for different symbols in figure, the second (typically 'GeneName') is used as prefered text for annotating the best points (if namesNBest allows to do so.)
annColor	(character or integer) colors for specific groups of annoatation (only if Mvalue is MArrayLM-object containing matrix $annot)
cexPt	(numeric) size of points, as expansion factor (see also cex in par)
cexSub	(numeric) size of subtitle, as expansion factor (see also cex in par)
cexTxLab	(numeric) size of text-labels for points, as expansion factor (see also cex in par)
namesNBest	(integer or character) number of best points to add names in figure; if 'passThr' all points passing FDR and FC-filtes will be selected; if the initial object Mvalue contains a list-element called 'annot' the second of the column specified in argument annotColumn will be used as text
NbestCol	(character or integer) colors for text-labels of best points
sortLeg	(character) sorting of 'SpecType' annotation either ascending ('ascend') or descending ('descend'), no sorting if NULL
NaSpecTypeAsContam	(logical) consider lines/proteins with NA in Mvalue$annot[,"SpecType"] as contaminants (if a 'SpecType' for contaminants already exits)

useMar	(numeric,length=4) custom margings (see also par)
returnData	(logical) optional returning data.frame with (ID, Mvalue, pValue, FDRvalue, passFilt)
callFrom	(character) allow easier tracking of message(s) produced
silent	(logical) suppress messages
debug	(logical) additional messages for debugging

Details

In high-throughput biology data are typically already transformed to log2 and thus, the 'M'-value represents a relative change. Besides, output from statistical testing by moderTest2grp or moderTestXgrp can be directly read to produce Volcano plots for diagnostic reasons. Please note, that plotting a very number of points in transparency (eg >10000) may take several seconds.

Value

MA-plot only

See Also

(for PCA) plotPCAw)

Examples

```
library(wrMisc)
set.seed(2005); mat <- matrix(round(runif(900),2), ncol=9)
rownames(mat) <- paste0(rep(letters[1:25],each=4), rep(letters[2:26],4))
mat[1:50,4:6] <- mat[1:50,4:6] + rep(c(-1,1)*0.1,25)
mat[3:7,4:9] <- mat[3:7,4:9] + 0.7
mat[11:15,1:6] <- mat[11:15,1:6] - 0.7
## assume 2 groups with 3 samples each
gr3 <- gl(3,3,labels=c("C","A","B"))
tRes2 <- moderTest2grp(mat[,1:6], gl(2,3), addResults = c("FDR","means"))
# Note: due to the small number of lines only FDR chosen to calculate
```

---

writeFasta2 *Write sequences in fasta format to file This function writes sequences from character vector as fasta formatted file (from* R*hrefhttps://www.uniprot.orgUniProt) Line-headers are based on names of elements of input vector* prot*. This function also allows comparing the main vector of sequences with a reference vector* ref *to check if any of the sequences therein are truncated.*

Description

Write sequences in fasta format to file

This function writes sequences from character vector as fasta formatted file (from UniProt) Line-headers are based on names of elements of input vector `prot`. This function also allows comparing the main vector of sequences with a reference vector `ref` to check if any of the sequences therein are truncated.

Usage

```
writeFasta2(
  prot,
  fileNa = NULL,
  ref = NULL,
  lineLength = 60,
  eol = "\n",
  truSuf = "_tru",
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
```

Arguments

`prot`	(character) vector of sequenes, names will be used for fasta-header
`fileNa`	(character) name (and path) for file to be written
`ref`	(character) optional/additional set of (reference-) sequences (only for comparison to `prot`), length of proteins from `prot` will be checked to mark truncated proteins by '_tru'
`lineLength`	(integer, length=1) number of sequence characters per line (default 60, should be >1 and <10000)
`eol`	(character) the character(s) to print at the end of each line (row); for example, eol = "\r\n" will produce Windows' line endings on a Unix-alike OS
`truSuf`	(character) suffix to be added for sequences found truncated when comparing with `ref`
`silent`	(logical) suppress messages
`debug`	(logical) supplemental messages for debugging
`callFrom`	(character) allows easier tracking of messages produced

Details

Sequences without any names will be given generic headers like protein01 ... etc.

Value

This function writes the sequences from `prot` as fasta formatted-file

See Also

readFasta2 for reading fasta, write.fasta from the package seqinr

Examples

```
#
prots <- c(SEQU1="ABCDEFGHIJKL", SEQU2="CDEFGHIJKLMNOP")
writeFasta2(prots, fileNa=file.path(tempdir(),"testWrite.fasta"), lineLength=6)
```

Index