Package 'wrProteo'

October 7, 2020

Version 1.1.4

Title Proteomics Data Analysis Functions

Author Wolfgang Raffelsberger [aut, cre]

Maintainer Wolfgang Raffelsberger <w.raffelsberger@unistra.fr>

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 (Schubert et al 2017 <doi:10.1038/nprot.2017.040>) 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.
Subsequently, ROC curves (Hand and Till 2001 <doi:10.1023/A:1010920819831>) can be constructed to compare multiple analysis approaches.

Depends R (>= 3.1.0)

Imports grDevices, graphics, limma, knitr, stats, rmarkdown, wrMisc

Suggests fdrtool, MASS, RColorBrewer, ROTS, R.utils, sm, utils, wrGraph

License GPL-3

Encoding UTF-8

VignetteBuilder knitr, rmarkdown

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2020-10-07 14:00:02 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

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()
```

`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,
  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
callFrom	(character) allows easier tracking of message(s) produced

Value

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(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))
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

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)
```

---

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

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
```

`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

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)
```

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

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

graphic 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

Impute random values to NAs in matrix based on replicates (neighbour) values

---

Description

It is assumed that NA-values appear the data when quantitation values are very low, as this appears eg in proteomics. Thus the remaining lowest values may be used to guide imputation. Here, groups of replicate samples (grouping defined via gr of columns of dat) are inspected for each line to gather NA-neighbour values. Eg, if a given line contains for a set of 4 replicates 2 NA-values, the remaining 2 non-NA-values will be considered as NA-neighbours. Then, this function replaces NA-values based the sub-population of all NA-neighbours (across all groups of replicates and all lines), assuming a Gaussian distribution. Indeed, in a number of experimental settings 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 (quantitative) measurments in proteomics and transcriptomics this is known to happen. So this function allows to model and subsequently replace all NA-values by Gaussian random values based on the characteristics of NA-neighbours in the same data-set. However, defining these characteristics (via the arguments avSdH and avSdL) may be very delicate and visual verification of the plots produced is highly encouraged ! If more than 300 NA-neighbours were detected, the imputation will be based on a more restricted sub-set of data with >1 NA values (ie via the argument avSdH). Optionally a histogram may be plotted showing the initial, imputed and final distribution to check if 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. Of course, all decisions to replace values do have a strong impact on further steps of data-analysis and should be performed with care. Please note, that no distinction is made if values seem totally absent (all values of given line and group) as NA or partially absent (mixture of NA and real quantitations). Thus, truly absent groups may be over-estimated.

Usage

```
matrixNAneighbourImpute(
  dat,
  gr,
  retnNA = TRUE,
  avSdH = c(0.18, 0.5),
  avSdL = c(0.1, 0.5),
  plotHist = TRUE,
  xLab = NULL,
  tit = NULL,
  addImputDetail = TRUE,
  seedNo = 2018,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

dat	(matrix or data.frame) main data (may contain NA)
gr	(character or factor) grouping of columns of 'dat', replicate association
retnNA	(logical) decide if NA values should be removed or retained
avSdH	(numerical,length=2) population characteristics 'high' (mean and sd) for >1 NA-neighbours (per line)
avSdL	(numerical,length=2) population characteristics 'low' (mean and sd) for >0 NA-neighbours
plotHist	(logical) decide if supplemental figure with histogram shoud be drawn
xLab	(character) label on x-axis on plot
tit	(character) title on plot
addImputDetail	(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 message(s) produced

Value

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

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
datT6b <- matrixNAneighbourImpute(datT6,gr=gl(2,3))
head(datT6b$data)
```

---

plotROC *Plot ROC curves*

Description

plotROC plots ROC curves based on results from summarizeForROC. Does not return any data, plot only. Allows printing simultaneously multiple ROC curves from different studies. Was made for special consideration of 3 species mix as in proteomics benchmark In the simplest case data were prepared using moderTest2grp

Usage

```
plotROC(
  dat,
  ...,
  useCol = 2:3,
  methNames = NULL,
  col = NULL,
  pch = 1,
  bg = NULL,
  tit = NULL,
  xlim = NULL,
  point05 = 0.05,
  pointSi = 0.85,
  nByMeth = NULL,
  speciesOrder = NULL,
  txtLoc = c(0.4, 0.3, 0.04),
  legCex = 0.72,
  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')
useCol	(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-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)
addSuplT	(logical) add text with information about precision,accuracy and FDR
silent	(logical) suppress messages
callFrom	(character) allows easier tracking of messages produced

Value

plot only

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

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 GeneName, OrganismName, ProteinName) 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 GeneName, OrganismName, ProteinName) in separate columns.

Usage

```
readFasta2(
  filename,
  delim = "|",
  databaseSign = c("sp", "tr", "generic", "gi"),
  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 afetr the '>' (typically specifying the data-base-origin), they will be excluded from the sequance-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 message(s) produced
debug	(logical) supplemental messages for debugging

Value

return (based on 'tableOut') simple character vector (of sequence) with Uniprot ID as name or matrix with columns: 'database','uniqueIdentifier','entryName','proteinName','sequence' and further columns depending on argument UniprSep

See Also

scan or read.fasta

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)
```

---

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

Description

Quantification results form MaxQuant can be read using this function and relevant information extracted. The final output is a list containing 3 elements: $annot, $abund and optional $quant, or returns data.frame with entire content of file if separateAnnot=FALSE. This function has been developed using MaxQuant version 1.6.10.x, the format of resulting file 'proteinGroups.txt' is typically well conserved.

Usage

```
readMaxQuantFile(
  path,
  fileName = "proteinGroups.txt",
  normalizeMeth = "median",
  quantCol = "LFQ.intensity",
  contamCol = "Potential.contaminant",
  uniqPepPat = "Razor...unique.peptides",
  refLi = NULL,
 extrColNames = c("Majority.protein.IDs", "Fasta.headers", "Number.of.proteins"),
  specPref = c(conta = "conta|CON_|LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
  tit = NULL,
  wex = 1.6,
  separateAnnot = TRUE,
  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)
normalizeMeth	(character) normalization method (will be sent to 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
uniqPepPat	(character, length=1) pattern to search for columns with unique (razor) peptides using grep, default set to read unique razor-peptides
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
tit	(character) custom title to plot
wex	(numeric) relative expansion factor of the violin in plot
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 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 message produced

Value

list with $annot, $raw for initial abundance values and $quant with final normalized quantitations, or returns data.frame with annot and quant 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 MaxQuant default name)
fiNa <- "proteinGroupsMaxQuantUps1.txt"
specPref1 <- c(conta="conta|CON_|LYSC_CHICK", mainSpecies="YEAST",spike="HUMAN_UPS")
dataMQ <- readMaxQuantFile(path1, file=fiNa, specPref=specPref1)
summary(dataMQ$quant)
matrixNAinspect(dataMQ$quant, gr=gl(3,3))
```

---

readPDExport *Read tabulated files imported from Thermo ProteomeDiscoverer*

Description

Quantification results form Thermo ProteomeDiscoverer 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. This function has been developed using MaxQuant version Thermo ProteomeDiscoverer2.4, the format of resulting file is typically well conserved.

Usage

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

```
  separateAnnot = TRUE,
  plotGraph = TRUE,
  tit = "Proteome Discoverer",
  graphTit = NULL,
  wex = 1.6,
  specPref = c(conta = "CON_|LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
  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)
read0asNA	(logical) decide if initial quntifications at 0 should be transformed to NA
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."))
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
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
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
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) 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',...)
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of message(s) produced

Value

list with $annot, $raw for initial/raw abundance values and $quant with final normalized quantitations, or returns data.frame with annot and quant if separateAnnot=FALSE

See Also

read.table, normalizeThis) , readProlineFile

Examples

```
path1 <- system.file("extdata", package="wrProteo")
fiNa <- "exampleProtDiscov1.txt"
dataPD <- readPDExport(file=fiNa, path=path1)
summary(dataPD$quant)
matrixNAinspect(dataPD$quant, gr=gl(2,3))
```

---

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

Description

Quantification results form MS-Angel and Proline Proline should be first saved via Excel or LibreOffice as csv or tabulated txt. Such files can be read by this function and relevant information be extracted. The final output is a list containing 3 elements: $annot, $abund and optional $quant, or returns data.frame with entire content of file if separateAnnot=FALSE. Note: There is no normalization by default since quite frequently data produced by Proline are already sufficiently normalized. In case of doubt the figure prouced using the argument plotGraph=TRUE may help judging if distribtions are aligned suffiently well.

Usage

```
readProlineFile(
  fileName,
  path = NULL,
  logConvert = TRUE,
  quantCol = "^abundance_",
 annotCol = c("accession", "description", "is_validated", "coverage", "X.sequences",
    "X.peptides", "protein_set.score"),
  refLi = NULL,
  separateAnnot = TRUE,
  plotGraph = TRUE,
  tit = NULL,
  graphTit = NULL,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

fileName	(character) name of file to read
path	(character) optional path (note: Windows backslash sould be protected or written as '/')
logConvert	(logical) convert numeric data as log2, will be placed in $quant
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) (character) exact col-names or if length=1 pattern to search among column-names for $annot
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'
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of message(s) produced

Value

list with $annot, $raw and optional $quant, or returns data.frame with entire content of file if separateAnnot=FALSE

See Also

read.table

Examples

```
path1 <- system.file("extdata", package="wrProteo")
fiNa <- "exampleProlineABC.csv"
dataABC <- readProlineFile(file.path(path1,fiNa))
summary(dataABC$quant)
matrixNAinspect(dataABC$quant, gr=as.factor(substr(colnames(dataABC$quant),1,1)))
```

`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

matrix, optionally the file 'exportFileNa' may be written

See Also

readUniProtExport, readPDExport, readMaxQuantFile,

Examples

```
path1 <- system.file("extdata", package="wrProteo")
gtfFi <- file.path(path1, "UCSC_hg38_chr11extr.gtf")
# 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

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")
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

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)
```

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,
  thr = NULL,
  tyThr = "BH",
  columnTest = 1,
  spec = c("H", "E", "S"),
  annotCol = "spec",
  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
thr	(numeric) threshold, if NULL a panel of 108 values will be used for calculating specifcity and sensitivity
tyThr	(character,length=1) type of test-result to be used for sensitivity and specificity calculations (eg 'BH','lfdr' or 'p.value'), must be list-element of 'test'
columnTest	(character or integer) only in case 'tyThr' is matrix (as typically the case after testRobustToNAimputation) : which column of 'test$tyThr' should be used as test-result

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
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

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, 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 message produced

Value

limma-type S3 object of class 'MArrayLM' which can be accessed; 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 *Test robust to NA-imputation*

Description

testRobustToNAimputation replaces NA values based on group neigbours (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(0.18, 0.5),
  avSdL = c(0.1, 0.5),
  plotHist = FALSE,
  xLab = NULL,
  tit = NULL,
  seedNo = 2018,
  nLoop = 20,
  lfdrInclude = TRUE,
  ROTSn = NULL,
  silent = 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	(numeric) population characteristics (mean and sd) for >0 NA neighbours 'low' (per line)
plotHist	(logical) additional histogram of original, imputed and resultant distribution (made using matrixNAneighbourImpute)
xLab	(character) custom x-axis label
tit	(character) custom title
seedNo	(integer) seed-value for normal random values
nLoop	(integer) number of runs of independent NA-imputation
lfdrInclude	(logical) include lfdr estimations (may cause warning message(s) concerning convergence if few too lines/proteins in dataset tested).
ROTSn	(integer) number of repeats by ROTS, if NULL ROTS will not be called
silent	(logical) suppress messages
callFrom	(character) allows easier tracking of messages produced

Value

limma-type S3 object of class 'MArrayLM' which can be accessed; 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=100,ROTSn=0,lfdr=FALSE)
names(PLtestR1)
```

Index