

Package: wrProteo (via r-universe)

July 23, 2026

Version 2.1.0

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](https://doi.org/10.1093/nar/gky1049)>) can be read with automatic parsing and multiple annotation types (like species origin, abbreviated gene names, etc) extracted. Initial results from multiple software for protein (and peptide) quantitation can be imported (to a common format): MaxQuant (Tyanova et al 2016 <[doi:10.1038/nprot.2016.136](https://doi.org/10.1038/nprot.2016.136)>), Dia-NN (Demichev et al 2020 <[doi:10.1038/s41592-019-0638-x](https://doi.org/10.1038/s41592-019-0638-x)>), Fragpipe (da Veiga et al 2020 <[doi:10.1038/s41592-020-0912-y](https://doi.org/10.1038/s41592-020-0912-y)>), ionbot (Degroev et al 2021 <[doi:10.1101/2021.07.02.450686](https://doi.org/10.1101/2021.07.02.450686)>), MassChroq (Valot et al 2011 <[doi:10.1002/pmic.201100120](https://doi.org/10.1002/pmic.201100120)>), OpenMS (Strauss et al 2021 <[doi:10.1038/nmeth.3959](https://doi.org/10.1038/nmeth.3959)>), ProteomeDiscoverer (Orsburn 2021 <[doi:10.3390/proteomes9010015](https://doi.org/10.3390/proteomes9010015)>), Proline (Bouyssie et al 2020 <[doi:10.1093/bioinformatics/btaa118](https://doi.org/10.1093/bioinformatics/btaa118)>), AlphaPept (preprint Strauss et al <[doi:10.1101/2021.07.23.453379](https://doi.org/10.1101/2021.07.23.453379)>) and Wombat-P (Bouyssie et al 2023 <[doi:10.1021/acs.jproteome.3c00636](https://doi.org/10.1021/acs.jproteome.3c00636)>). Meta-data provided by initial analysis software and/or in sdrf format can be integrated to the analysis. Quantitative proteomics measurements frequently contain multiple NA values, due to physical absence of given peptides in some samples, limitations in sensitivity or other reasons. Help is provided to inspect the data graphically to investigate the nature of NA-values via their respective replicate measurements and to help/confirm the choice of NA-replacement algorithms. Meta-data in sdrf-format (Perez-Riverol et al 2020 <[doi:10.1021/acs.jproteome.0c00376](https://doi.org/10.1021/acs.jproteome.0c00376)>) or similar tabular formats can be imported and included. Missing values can be inspected and imputed based on the concept of NA-neighbours or other

methods. Dedicated filtering and statistical testing using the framework of package 'limma' ([doi:10.18129/B9.bioc.limma](https://doi.org/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](https://doi.org/10.1038/nbt.3685), Ramus et al 2016 [doi:10.1016/j.jprot.2015.11.011](https://doi.org/10.1016/j.jprot.2015.11.011)), can be run with special options considering such sub-groups during normalization and testing. Subsequently, ROC curves (Hand and Till 2001 [doi:10.1023/A:1010920819831](https://doi.org/10.1023/A:1010920819831)) can be constructed to compare multiple analysis approaches. As detailed example the data-set from Ramus et al 2016 ([doi:10.1016/j.jprot.2015.11.011](https://doi.org/10.1016/j.jprot.2015.11.011)) quantified by MaxQuant, ProteomeDiscoverer, and Proline is provided with a detailed analysis of heterologous spike-in proteins.

Depends R (>= 3.5.0)

Imports grDevices, graphics, knitr, limma, stats, utils, wrMisc (>= 2.1.0)

Suggests data.table, fdrtool, kableExtra, MASS, RColorBrewer, readxl, ROTS, rmarkdown, R.utils, sm, wrGraph (>= 1.3.16)

License GPL-3

Encoding UTF-8

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Repository <https://cran.r-universe.dev>

Date/Publication 2026-07-23 18:40:02 UTC

RemoteUrl <https://github.com/cran/wrProteo>

RemoteRef HEAD

RemoteSha aaa0630bf4a92dd533b6efb2af6d2fa1db2df4d0

Contents

.atomicMasses	4
.checkKnitrProt	4
.checkSetupGroups	5
.commonSpecies	6
.extrSpecPref	7
.getSpeciesFromEntryName	8
.imputeNA	9
.parseFastaHeader	10
.plotQuantDistr	11
AAmass	12
AucROC	13

cleanListCoNames	14
combineMultFilterNAimput	15
convAASeq2mass	18
corColumnOrder	19
countNoOfCommonPeptides	20
exportAsWombatP	21
exportSdrfDraft	23
extractTestingResults	24
extrSpeciesAnnot	27
foldChangeArrow2	28
fuseProteomicsProjects	29
getUPS1acc	31
inspectSpeciesIndic	32
isolNAneighb	33
massDeFormula	34
matrixNAinspect	35
matrixNAneighbourImpute	36
plotROC	39
razorNoFilter	41
readAlphaPeptFile	42
readDiaNNFile	45
readDiaNNPeptides	48
readFasta2	50
readFragpipeFile	52
readIonbotPeptides	54
readMassChroQFile	57
readMaxQuantFile	59
readMaxQuantPeptides	63
readOpenMSFile	66
readProlineFile	68
readProtDiscovererPeptides	71
readProtDiscovFile	71
readProteomeDiscovererFile	74
readProteomeDiscovererPeptides	77
readSampleMetaData	79
readSdrf	82
readUCSCtable	83
readUniProtExport	84
readWombatNormFile	86
removeSampleInList	89
replMissingProtNames	90
shortSoftwName	91
summarizeForROC	92
test2grp	95
testRobustToNAimputation	96
VolcanoPlotW2	99
writeFasta2	102

*.atomicMasses**Molecular mass for Elements*

Description

This fuction returns the molecular mass based of main elements found in biology/proteomics as average and mono-isotopic mass. The result includes H, C, N, O, P, S, Se and the electrone. The values are bsd on <http://www.ionsource.com/Card/Mass/mass.htm> in ref to <http://physics.nist.gov/Comp> (as of 2019).

Usage

```
.atomicMasses()
```

Value

This function returns a numeric matrix with mass values

See Also

[massDeFormula](#)

Examples

```
.atomicMasses()
```

*.checkKnitrProt**Checking presence of knitr and rmarkdown*

Description

This function allows checking presence of knitr and rmarkdown

Usage

```
.checkKnitrProt(tryF = FALSE)
```

Arguments

tryF (logical)

Value

This function returns a logical value

See Also[presenceFilt](#)**Examples**

```
.checkKnitrProt()
```

<code>.checkSetupGroups</code>	<i>Additional/final Check And Adjustments To Sample-order After read-SampleMetaData()</i>
--------------------------------	---

Description

This (low-level) function performs an additional/final check & adjustments to sample-names after `readSampleMetaData()`

Usage

```
.checkSetupGroups(  
  abund,  
  setupSd,  
  gr = NULL,  
  sampleNames = NULL,  
  quantMeth = NULL,  
  silent = FALSE,  
  callFrom = NULL,  
  debug = FALSE  
)
```

Arguments

<code>abund</code>	(matrix or data.frame) abundance data, only the colnames will be used
<code>setupSd</code>	(list) describing sammples-setup, typically produced by <code>readSampleMetaData()</code> (from this package)
<code>gr</code>	(factor) optional custom information about replicate-layout, has priority over <code>setupSd</code>
<code>sampleNames</code>	(character) custom sample-names, has priority over <code>abund</code> and <code>setuoSd</code>
<code>quantMeth</code>	(character) 2-letter abbreviation of name of quantitation-software (eg 'MQ')
<code>silent</code>	(logical) suppress messages
<code>callFrom</code>	(character) allow easier tracking of messages produced
<code>debug</code>	(logical) display additional messages for debugging

Details

In particular, when no meta-data or expriental setup was found this function tries to get a min of information

Value

This function returns an enlarged/updated list 'setupSd' (set setupSd\$sampleNames, setupSd\$groups)

See Also

used in readProtDiscovererFile, [readMaxQuantFile](#), [readProlineFile](#), [readFragpipeFile](#)

Examples

```
abun1 <- matrix(1:16, ncol=8, dimnames=list(NULL, paste("samp", LETTERS[8:1], sep="_")))
sdrf1 <- data.frame(source.name=paste(rep(LETTERS[1:4], each=2), 1:2, sep="_"),
  assay.name=paste0("run", 1:8), comment.data.file=paste0("MSrun", 8:1))
setU1 <- list(level=gl(4,2), meth="lowest", sampleNames=paste("samp", LETTERS[1:8], sep="_"),
  sdrfDat=sdrf1, annotBySoft=NULL)
.checkSetupGroups(abun1, setU1)
```

.commonSpecies

Get Matrix With UniProt Abbreviations For Selected Species As Well As Simple Names

Description

This (low-level) function allows accessing matrix with UniProt abbreviations for species frequently used in research. This information may be used to harmonize species descriptions or extract species information out of protein-names.

Usage

```
.commonSpecies()
```

Value

This function returns a 3-column matrix with species names and UniProt-extensions

See Also

used eg in readFasta2, readProtDiscovererFile, [readMaxQuantFile](#), [readProlineFile](#), [readFragpipeFile](#)

Examples

```
.commonSpecies()
```

.extrSpecPref	<i>Extract Additional Information To Construct The Colum 'SpecType', Allows Adding Information From Fasta</i>
---------------	---

Description

This (low-level) function creates the column `annot[, 'SpecType']` which may help distinguishing different lines/proteins. This information may, for example, be used to normalize only to all proteins of a common background matrix (species). In order to compare `specPref` a species-column will be added to the annotation (`annot`) - if not already present. If `$mainSpecies` or `$conta`: match to `annot[, "Species"]`, `annot[, "EntryName"]`, `annot[, "GeneName"]`, if `length==1` `grep` in `annot[, "Species"]`

Usage

```
.extrSpecPref(
  specPref,
  annot,
  useColumn = c("Species", "EntryName", "GeneName", "Accession"),
  suplInp = NULL,
  soft = NA,
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
```

Arguments

<code>specPref</code>	(list) may contain <code>\$mainSpecies</code> , <code>\$conta</code> ...
<code>annot</code>	(matrix) main protein annotation
<code>useColumn</code>	(factor) columns from <code>annot</code> to use/mine
<code>suplInp</code>	(matrix) additional custom annotation
<code>soft</code>	(character, length=1) additional info which software was initially used (so far only special treatment for IB)
<code>silent</code>	(logical) suppress messages
<code>debug</code>	(logical) display additional messages for debugging (starting with 'mainSpecies', 'conta' and others - later may overwrite prev settings)
<code>callFrom</code>	(character) allow easier tracking of messages produced

Details

Different to `readSampleMetaData` this function also considers the main annotation as extracted with main quantification data. For example, this function can complement protein annotation data if columns 'Accession', 'EntryName' or 'SpecType' are missing

Value

This function returns a matrix with additional column 'SpecType'

See Also

used in [readProtDiscovererFile](#), [readMaxQuantFile](#), [readProlineFile](#), [readFragpipeFile](#)

Examples

```
annot1 <- cbind( Leading.razor.protein=c("sp|P00925|EN02_YEAST",
    "sp|Q3E792|RS25A_YEAST", "sp|P09938|RIR2_YEAST", "sp|P09938|RIR2_YEAST",
    "sp|Q99186|AP2M_YEAST", "sp|P00915|CAH1_HUMAN"),
    Species= rep(c("Saccharomyces cerevisiae", "Homo sapiens"), c(5,1)))
specPref1 <- list(conta="CON_|LYSC_CHICK",
    mainSpecies="OS=Saccharomyces cerevisiae", spike="P00915") # MQ type
.extrSpecPref(specPref1, annot1, useColumn=c("Species", "Leading.razor.protein"))
```

[.getSpeciesFromEntryName](#)

Complete/Extract Species Information Based On EntryName

Description

This (low-level) function allows completing species information when standard species Entry-Names are given. Eg, for the EntryName 'RHG10_HUMAN' it will be concluded that this is Homo Sapiens

Usage

```
.getSpeciesFromEntryName(x, returnAs = 2)
```

Arguments

x	(matrix or data.frame) annotation table with columns 'EntryName' and 'Species'
returnAs	(integer) refers to which column of .commonSpecies() should be returned : use 2 for full scientific names (eg Homo sapiens), use 3 for short names (eg Human)

Value

This function returns a matrix of dimensions as 'x', the column 'Species' will be now enriched/completed

See Also

[.commonSpecies](#), used in [readDiaNNFile](#)

Examples

```
anno1 <- cbind(EntryName=c("RHG10_HUMAN", "abc_unknown", "HA22_MOUSE"), Species=rep(NA, 3))
.getSpeciesFromEntryName(anno1)
```

<code>.imputeNA</code>	<i>Basic NA-imputaton (main)</i>
------------------------	----------------------------------

Description

This (lower-level) function allows to perform the basic NA-imputaton. Note, at this point the information from argument `gr` is not used.

Usage

```
.imputeNA(  
  dat,  
  gr = NULL,  
  impParam,  
  exclNeg = TRUE,  
  inclLowValMod = TRUE,  
  silent = FALSE,  
  debug = FALSE,  
  callFrom = NULL  
)
```

Arguments

<code>dat</code>	(matrix or data.frame) main data (may contain NA)
<code>gr</code>	(character or factor) grouping of columns of <code>dat</code> , replicate association
<code>impParam</code>	(numeric) 1st for mean; 2nd for sd; 3rd for seed
<code>exclNeg</code>	(logical) exclude negative
<code>inclLowValMod</code>	(logical) label on x-axis on plot
<code>silent</code>	(logical) suppress messages
<code>debug</code>	(logical) supplemental messages for debugging
<code>callFrom</code>	(character) allow easier tracking of messages produced

Value

This function returns a list with `$data` and `$datImp`

See Also

for more complex treatment [matrixNAneighbourImpute](#);

Examples

```
dat1 <- matrix(11:22, ncol=4)  
dat1[3:4] <- NA  
.imputeNA(dat1, impParam=c(mean(dat1, na.rm=TRUE), 0.1))
```

<code>.parseFastaHeader</code>	<i>Parse Fasta Header</i>
--------------------------------	---------------------------

Description

Parse fasta header (from [UniProt](#)) to extract different annotation fields

Usage

```
.parseFastaHeader(
  header,
  delim = "|",
  databaseSign = c("sp", "tr", "generic", "conta", "synt", "gi"),
  UniprSep = c("OS=", "OX=", "GN=", "PE=", "SV="),
  asList = FALSE,
  silent = FALSE,
  callFrom = NULL,
  debug = FALSE
)
```

Arguments

<code>header</code>	(character) fasta-header
<code>delim</code>	(character) delimiter (ie primary separator)
<code>databaseSign</code>	(character) characters at beginning right after the '>' (typically specifying the data-base-origin), they will be excluded from the sequence-header
<code>UniprSep</code>	(character) separators for further separating entry-fields if <code>tableOut=TRUE</code> ; with these delimiter fields a space is assumed in addition to the separators; see also UniProt-FASTA-headers
<code>asList</code>	(logical) if <code>asList=TRUE</code> , the function returns a list with two matrixes, one for primary parsing and another matrix for further parsing (using <code>UniprSep</code>), otherwise all will be combined in single matrix
<code>silent</code>	(logical) suppress messages
<code>callFrom</code>	(character) allows easier tracking of messages produced
<code>debug</code>	(logical) supplemental messages for debugging

Value

This function returns (depending on argument `asList`) a) a matrix with columns: `'db'`, `'uniqueIdentifier'`, `'entryName'`, `'protein'` and further columns depending on argument `UniprSep` of b) a list with matrix of primary parsing (argument `delim`) and matrix from further parsing (argument `UniprSep`)

See Also

This function is use by [readFasta2](#), [writeFasta2](#) for writing as fasta; for reading [readLines](#) or `read.fasta` from the package [seqinr](#)

Examples

```
.parseFastaHeader(">sp|P00760|TRY1_BOVIN Serine protease 1 OS=Bos taurus OX=9913 GN=PRSS1 PE=1")
```

<code>.plotQuantDistr</code>	<i>Generic Plotting Of Density Distribution For Quantitation Import-functions</i>
------------------------------	---

Description

This (low-level) function allows (generic) plotting of density distribution for quantitation import-functions

Usage

```
.plotQuantDistr(  
  abund,  
  quant,  
  custLay = NULL,  
  normalizeMeth = NULL,  
  softNa = NULL,  
  refLi = NULL,  
  refLiIni = NULL,  
  notLogAbund = NA,  
  figMarg = c(3.5, 3.5, 3, 1),  
  col = "rainbow",  
  tit = NULL,  
  las = NULL,  
  cexAxis = 0.8,  
  nameSer = NULL,  
  cexNameSer = NULL,  
  silent = FALSE,  
  callFrom = NULL,  
  debug = FALSE  
)
```

Arguments

<code>abund</code>	(matrix or data.frame) abundance data, will be plotted as distribution
<code>quant</code>	(matrix or data.frame) optional additional abundance data, to plot 2nd distribution, eg of normalized data
<code>custLay</code>	(matrix) describing sample-setup, typically produced by
<code>normalizeMeth</code>	(character, length=1) name of normalization method (will be displayed in title of figure)
<code>softNa</code>	(character, length=1) name of quantitation-software (typically 2-letter abbreviation, eg 'MQ')

refLi	(integer) to display number reference lines
refLiIni	(integer) to display initial number reference lines
notLogAbund	(logical) set to TRUE if abund is linear but should be plotted as log2
figMarg	(numeric, length=4) custom figure margins (will be passed to par), defaults to <code>c(3.5, 3.5, 3, 1)</code>
col	(character) custom colors (length must match number of columns)
tit	(character) custom title
las	(integer) indicate orientation of text in axes
cexAxis	(numeric) size of numeric axis labels as cex-expansion factor (see also par)
nameSer	(character) custom label for data-sets or columns (length must match number of data-sets)
cexNameSer	(numeric) size of individual data-series labels as cex-expansion factor (see also par)
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of messages produced
debug	(logical) display additional messages for debugging

Value

This function returns logical value (if data were valid for plotting) and produces a density distribution figure (if data were found valid)

See Also

used in [readProtDiscovererFile](#), [readMaxQuantFile](#), [readProlineFile](#), [readFragpipeFile](#)

Examples

```
set.seed(2018); datT8 <- matrix(round(rnorm(800) +3,1), nc=8, dimnames=list(paste(
  "l1",1:100,sep=""), paste(rep(LETTERS[1:3],c(3,3,2)),letters[18:25],sep="")))
.plotQuantDistr(datT8, quant=NULL, refLi=NULL, tit="Synthetic Data Distribution")
```

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 H2O 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 from 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"),
  returnIfInvalid = NA,
  silent = FALSE,
  debug = 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
returnIfInvalid	(NA or NULL) what to return if data for calculating ROC is invalid or incomplete

silent (logical) suppress messages
 debug (logical) additional messages for debugging
 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

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](#), etc.

Examples

```
set.seed(2019); test1 <- list(annot=cbind(Species=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"), annotCol="Species")
AucROC(roc1)
```

cleanListCoNames

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

Description

This function allows to manipulate sample-names (ie colnames of abundance data) in a batch-wise manner from data stored as multiple matrixes or data.frames of a list. 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/substituted (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"),
  mathOper = NULL,
  silent = FALSE,
```

```

    debug = 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 performed 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
mathOper	(character, length=1) optional mathematical operation on numerical part of sample-names (eg mathOper=' / 2 ' for deviding numeric part of colnames by 2)
silent	(logical) suppress messages
debug	(logical) additional messages for debugging
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(row=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. This function takes both the original data and the NA-imputed data to create several different filters (number of NAs, min abundance, ..) and finally combines them. The filter-component aiming to take away the least abundant values (using the imputed data) can be fine-tuned by the argument abundThr.

Usage

```

combineMultFilterNAimput(
  dat,
  imputed,
  grp,
  useComparison = "all",
  experSetup = NULL,
  annDat = NULL,
  abundThr = NULL,
  colRazNa = NULL,
  colTotNa = NULL,
  minSpeNo = 1,
  minTotNo = 2,
  maxGrpMiss = 1,
  ratMaxNA = 0.8,
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)

```

Arguments

<code>dat</code>	(matrix or data.frame) main data (may contain NA)
<code>imputed</code>	(character) same as 'dat' but with all NA imputed
<code>grp</code>	(character or factor) define groups of replicates (in columns of 'dat')
<code>useComparison</code>	(character or matrix) argument allowing to specify which pairwise comparisons should be performed, default <code>useComparison=NULL</code> will run all pairwise comparisons; may be character as result of combining two group-names (from argument <code>grp</code>) (eg 'A-B', beware, the separator may not appear inside group-names) or 2-column matrix of group-names (thus, without separator; names combined with separator may be shown as rownames) or 2-column matrix of indexes to sorted group-names; default setting of <code>useComparison='all'</code> will select all possible pairwise combinations
<code>experSetup</code>	(list) optional experimental setup (list with <code>\$ind</code> , <code>\$pwGrpIndex</code> , <code>\$sep</code> and <code>\$pwGrpNa</code> as obtained using <code>getPairwiseSetup</code>)
<code>annDat</code>	(matrix or data.frame) annotation data (should match lines of 'dat')
<code>abundThr</code>	(numeric) optional threshold filter for minimum abundance
<code>colRazNa</code>	(character) if razor peptides are used: column name for razor peptide count
<code>colTotNa</code>	(character) column name for total peptide count
<code>minSpeNo</code>	(integer) minimum number of specific peptides for maintaining proteins
<code>minTotNo</code>	(integer) minimum total ie max razor number of peptides
<code>maxGrpMiss</code>	(numeric) passed to presenceFilt : at least 1 group has not more than this number of NAs (otherwise mark line as bad)
<code>ratMaxNA</code>	(numeric) passed to presenceFilt : at least 1 group below this content of NA values

silent	(logical) suppress messages
debug	(logical) additional messages for debugging
callFrom	(character) allows easier tracking of messages produced

Details

The main filtering step compares the means for each group and line (from argument `imputed`), at least one group-mean has to be superior the threshold (`abundThr`, based on hypothesis that if all conditions represent extrememly low measures their differential may not be determined with certainty).

The filter addressing the number of missing values (NA) uses the original data (argument `dat`), the arguments `colTotNa`, `minSpeNo` and `minTotNo` are used during 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.

Value

This function returns a list with `$data` (unmodified input from `dat`), `$nNA`, `$randParam`, `$NAnei-gLst`, `$seed`, `$filt` (the reommended filtering : TRUE for keeping value), and `$annot`

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)
head(datT6c$filt)
## with custom choice of comparisons :
gr2 <- gl(2, 3, labels=c("A","B"))
datT6d <- combineMultFilterNAimput(datT6, datT6b, grp=gr2, useCom="B-A", abundThr=2)
head(datT6d$filt)
```

convAASeq2mass	<i>Molecular mass for amino-acids</i>
----------------	---------------------------------------

Description

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

Usage

```
convAASeq2mass(
  x,
  massTy = "mono",
  seqName = TRUE,
  silent = FALSE,
  debug = 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
debug	(logical) additional messages for debugging
callFrom	(character) allows easier tracking of messages produced

Value

This function returns a vector with masses for all amino-acids (argument 'massTy' to switch from 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, Data.frames And Vectors***Description**

This function orders columns in list of matrixes (or matrix) according to argument `sampNames` and also offers an option for changing names of columns. It was (initially) designed to adjust/correct the order of samples after import using `readMaxQuantFile()`, `readProteomeDiscovererFile()` etc. The input may also be `MArrayLM`-type object from package **limma** or from functions `moderTestXgrp` or `moderTest2grp`.

Usage

```
corColumnOrder(
  dat,
  sampNames,
  replNames = NULL,
  useListElem = c("quant", "raw", "counts"),
  annotElem = "sampleSetup",
  newNames = NULL,
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
```

Arguments

<code>dat</code>	(matrix, list or <code>MArrayLM</code> -object from <code>limma</code>) main input of which columns should get re-ordered, may be output from <code>moderTestXgrp</code> or <code>moderTest2grp</code> .
<code>sampNames</code>	(character) column-names in desired order for output (its content must match <code>colnames</code> of <code>dat</code> or <code>replNames</code> , if used)
<code>replNames</code>	(character) option for replacing column-names by new/different <code>colnames</code> ; should be vector of NEW column-names (in order as input from <code>dat</code> !), allows renaming <code>colnames</code> before defining new order
<code>useListElem</code>	(character) in case <code>dat</code> is list, all list-elements who's columns should get (re-)ordered
<code>annotElem</code>	(character) name of list-element of <code>dat</code> with annotation data to get in new order
<code>newNames</code>	deprecated, please use <code>replNames</code> instead
<code>silent</code>	(logical) suppress messages
<code>debug</code>	(logical) display additional messages for debugging
<code>callFrom</code>	(character) allows easier tracking of messages produced

Value

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

See Also

[readMaxQuantFile](#), [readProteomeDiscovererFile](#); [moderTestXgrp](#) or [moderTest2grp](#)

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])

dat2 <- list(quant=dat1, raw=dat1)
dat2
corColumnOrder(dat2, sampNames=LETTERS[1:5])
corColumnOrder(dat2, sampNames=LETTERS[1:5], rep1Names=c("Dd","Aa","Cc","Ee","Bb"))
```

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 [Rhrefhttps://bioconductor.org/packages/release/bioc/html/cleaver.html](https://bioconductor.org/packages/release/bioc/html/cleaver.html)cleaver. 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,
  debug = FALSE,
  callFrom = NULL
)
```

Arguments

...	(list) multiple lists of (ini-silico) digested proteins (typically protein ID as names) with their respective 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
debug	(logical) display additional messages for debugging
callFrom	(character) allow easier tracking of message(s) produced

Value

This function returns a list with \$byPep as list of logical matrixes for each peptide (as lines) 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
```

exportAsWombatP

Export As Wombat-P Set Of Files

Description

This function allows exporting objects created from wrProteo to the format of Wombat-P **Wombat-P**.

Usage

```
exportAsWombatP(
  wrProtObj,
  path = ".",
  combineFractions = "mean",
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
```

Arguments

<code>wrProtObj</code>	(list produced by any import-function from <code>wrProteo</code>) object which will be exported as Wombat-P format
<code>path</code>	(character) the location where the data should be exorted to
<code>combineFractions</code>	(NULL or character (length=1)) if not NULL this assigns the method how multiple farctions should be combined (at this point only the method 'mean' is implemented)
<code>silent</code>	(logical) suppress messages
<code>debug</code>	(logical) display additional messages for debugging
<code>callFrom</code>	(character) allows easier tracking of messages produced

Value

This function creates a set of files (README.md, test_params.yml), plus a sud-directory containig file(s) (stand_prot_quant_method.csv); finally the function returns (NULL),

See Also

[readMaxQuantFile](#), [readProteomeDiscovererFile](#); [moderTestXgrp](#) or [moderTest2grp](#)

Examples

```
path1 <- system.file("extdata", package="wrProteo")
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")

exportAsWombatP(dataMQ, path=tempdir())
```

exportSdrfDraft

Export Sample Meta-data from Quantification-Software as Sdrf-draft

Description

Sample/experimental annotation meta-data from **MaxQuant** that was previously import can now be formatted in sdrf-style and exported using this function to write a draft-sdrf-file. Please note that this information will not `_complete_` in respect to all information used in data-bases like **Pride**. Sdrf-files provide additional meta-information about samles and MS-runs in a standardized format, they may also be part of submissions to **Pride**.

Usage

```
exportSdrfDraft(
  lst,
  fileName = "sdrfDraft.tsv",
  correctFileExtension = TRUE,
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
```

Arguments

<code>lst</code>	(list) object created by import-function (MaxQuant)
<code>fileName</code>	(character) file-name (and path) to be used when expirting
<code>correctFileExtension</code>	(logical) if TRUE the fileName will get a .tsv-extension if not already present
<code>silent</code>	(logical) suppress messages
<code>debug</code>	(logical) additional messages for debugging
<code>callFrom</code>	(character) allow easier tracking of messages produced

Details

Gathering as much as possible information about samples and MS-runs requires that the additional files created from software, like **MaxQuant** using [readMaxQuantFile](#), is present and was imported when calling the import-function (eg using the argument `_suplAnnotFile=TRUE_`). Please note that this functionality was designed for the case where no (external) sdrf-file is available. Thus, when data was imported including exteranl sdrf (uinsg the `_sdrf=_` argument), exporting incomplete annotation-data from **MaxQuant**-produced files does not make any sense and therefore won't be possible.

After exporting the draft sdrf the user is advised to check and complete the information in the resulting file. Unfortunately, not all information present in a standard sdrf-file (like on **Pride**) cannot be gathered automatically, but key columns are already present and thus may facilitate completing. Please note, that the file-format has been defined as .tsv, thus columns/fields should be separated

by tabs. At manual editing and completion, some editing- or tabulator-software may change the file-extension to .tsv.txt, in this case the final files should be renamed as .tsv to remain compatible with Pride.

At this point only the import of data from MaxQuant via [readMaxQuantFile](#) has been developed to extract information for creating a draft-sdrf. Other data/file-import functions may be further developed to gather as much as possible equivalent information in the future.

Value

This function writes an Sdrf draft to file

See Also

This function may be used after reading/importing data by [readMaxQuantFile](#) in absence of sdrf

Examples

```
path1 <- system.file("extdata", package="wrProteo")
fiNaMQ <- "proteinGroups.txt.gz"
dataMQ <- readMaxQuantFile(path1, file=fiNaMQ, refLi="mainSpe", sdrf=FALSE, suplAnnotFile=TRUE)
## Here we'll write simply in the current temporary directory of this R-session
exportSdrfDraft(dataMQ, file.path(tempdir(), "testSdrf.tsv"))
```

extractTestingResults *Extract Results From Moderated t-tests*

Description

This function allows convenient access to results produced using the functions [moderTestXgrp](#) or [moderTest2grp](#). The user can define the threshold and which type of multiple testing correction should be used (as long as the multiple testing correction method cited 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", "ProteinName"),
  nSign = 6,
  outTy = "slim",
  sep = NULL,
  addTy = NULL,
  sortBy = NULL,
```



```

    rowNames = TRUE,
    filename = NULL,
    fileTy = "csvUS",
    silent = FALSE,
    debug = FALSE,
    callFrom = NULL
)

```

Arguments

stat	('MArrayLM'-object or list) designed for the output from moderTest2grp or moderTestXgrp
compNo	(integer) the comparison name/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 statistical 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
outTy	(character) additional groups of columns to add ("allMeans", "all" or "slim") in addition; if outTy="slim" only the FDR- and FC-columns directly concerned will be extracted/displayed, column-names will be trimmed to max 26 characters; if outTy="slim2" only the FDR- and FC-columns directly concerned will be extracted/displayed, FDR- and FC- columns will have generic names (without indicating the comparison), all text will be trimmed to max 38 characters; if outTy="slim3" like slim2 but without ProteinName
sep	(character of length=1 or NULL) custom choice for separator for column-names of pairwise comparisons, otherwise based on stat or (if outTy="slim3" or outTy="slim4") '_vs_' (or '__vs__' if '-' or '_' appear in groups)
addTy	this argument has been deprecated and replaced by outTy (content of addTy will be ignored)
sortBy	(character or logical) optional sorting of results according to argument compNo, may be name of mutiple-testing correction (eg 'BH', see also argument statTy), if sortBy=TRUE the first column of argument statTy will be used
rowNames	(character or logical) optional custom rownames, or if rowNames=FALSE no row-names
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
debug	(logical) display additional messages for debugging
callFrom	(character) allow easier tracking of messages produced

Details

One single given comparison has to be selected by the user via argument `compNo`, multiple options exist : 1) The user may give an index which refers to the *i*'th comparison stored in the object `stat$setup` 2) It may be more convenient to directly cite a given comparison (eg `compNo=c("B-C")`) in the examples below). Such comparisons may be 'inversed' to the ones originally performed, the M-values will be adjusted accordingly (eg `compNo=c("C-B")`) 3) It is also possible to give a vector of 2 integers which will be used as index to the levels of the initial group-assignment, beware that levels are sorted.

The number and type of columns returned can be modulated using the argument `outTy`. `outTy='slim'` gives the fewer columns, only FDR and FC of current comparison specified in argument `compNo` will be returned. With `outTy='slim2'` the column-names for statistical testing and log Fold-Change will be generic, all colnames will be trimmed to max 25 characters `outTy='all'` results in a rather exhaustive collection of columns (including annotation data, if present), The option `outTy='slim3'` is like `slim2` but without column `ProteinName` while default `outTy='allMeans'` gives a compromise for the number of columns returned.

Value

This function returns a data.frame or matrix (if no annotation added) with values (and annotation) conform to filter criteria

See Also

[testRobustToNAimputation](#), [moderTestXgrp](#) or [moderTest2grp](#)

Examples

```
grp <- factor(rep(LETTERS[4:2],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)
extractTestingResults(test8, compNo=c("B-C"))

extractTestingResults(test8, outTy="all")
extractTestingResults(test8, outTy="slim")
```

extrSpeciesAnnot	<i>Extract species annotation</i>
------------------	-----------------------------------

Description

extrSpeciesAnnot identifies species-related annotation (as suffix to identifiers) for data containing multiple species and returns alternative (short) names. This function also suppresses extra heading or trailing 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,  
  debug = 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 length must fit to argument annot
silent	(logical) suppress messages
debug	(logical) display additional messages for debugging
callFrom	(character) allow 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	<i>Add arrow for expected Fold-Change to VolcanoPlot or MA-plot</i>
------------------	---

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.

Usage

```
foldChangeArrow2(
  FC,
  useComp = 1,
  isLin = TRUE,
  asX = TRUE,
  col = 1,
  arr = c(0.005, 0.15),
  lwd = NULL,
  addText = c(line = -0.9, cex = 0.7, txt = "expected", loc = "toright"),
  returnRatio = FALSE,
  silent = FALSE,
  debug = 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) indicate 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
debug	(logical) display additional messages for debugging
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 `elems` `side` and `adjust` will be redefined to accomodate the text in the corner specified.

Ultimately this function will be integrated 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

```
#deprecated# foldChangeArrow2(FC=1.5)
```

`fuseProteomicsProjects`

Combine Multiple Proteomics Data-Sets

Description

This function allows combining up to 3 separate data-sets previously imported using `wrProteo`.

Usage

```
fuseProteomicsProjects(  
  x,  
  y,  
  z = NULL,  
  columnNa = "Accession",  
  NA.rm = TRUE,  
  listNa = c(quant = "quant", annot = "annot"),  
  all = FALSE,  
  textModif = NULL,  
  shortNa = NULL,  
  retProtLst = FALSE,  
  silent = FALSE,  
  debug = FALSE,  
  callFrom = NULL  
)
```

Arguments

x	(list) First Proteomics data-set
y	(list) Second Proteomics data-set
z	(list) optional third Proteomics data-set
columnNa	(character) column names from annotation
NA.rm	(logical) remove NAs
listNa	(character) names of key list-elements from x to be treated; the first one is used as pattern for the format of quantitation data, , the last one for the annotation data
all	(logical) union of intersect or merge should be performed between x, y and z
textModif	(character) Additional modifications to the identifiers from argument columnNa; so far integrated: rmPrecAA for removing preceding caps letters (amino-acids, eg [KR].AGVIFPVGR.[ML] => AGVIFPVGR) or rmTerminalDigit for removing terminal digits (charge-states)
shortNa	(character) for appending to output-colnames
retProtLst	(logical) return list-object similar to input, otherwise a matrix of fused/aligned quantitation data
silent	(logical) suppress messages
debug	(logical) additional messages for debugging
callFrom	(character) allow easier tracking of messages produced

Details

Some quantification software may give some identifiers multiple times, ie as multiple lines (eg for different modifications or charge states, etc). In this case this function tries first to summarize all lines with identical identifiers (using the function [combineRedundLinesInList](#) which used by default the median value). Thus, it is very important to know your data and to understand when lines that appear with the same identifiers should/may be fused/summarized without doing damage to the later biological interpretation ! The user may specify for each dataset the column out of the protein/peptide-annotation to use via the argument columnNa. Then, this content will be matched as identical match, so when combining data from different software special care should be taken !

Please note, that (at this point) the data from different series/objects will be joined as they are, ie without any additional normalization. It is up to the user to inspect the resulting data and to decide if and which type of normalization may be suitable !

Please do NOT try combining protein and peptide quantification data.

Value

This function returns a list with the same number of list-elements as \$x, ie typically this contains : \$raw (initial/raw abundance values), \$quant with final normalized quantitations, \$annot, optionally \$counts an array with number of peptides, \$quantNotes or \$notes

See Also

[sd](#)

Examples

```
path1 <- system.file("extdata", package="wrProteo")
dataMQ <- readMaxQuantFile(path1, specPref=NULL, normalizeMeth="median")
MCproFi1 <- "tinyMC.RData"
dataMC <- readMassChroQFile(path1, file=MCproFi1, plotGraph=FALSE)
dataFused <- fuseProteomicsProjects(dataMQ, dataMC)
dim(dataMQ$quant)
dim(dataMC$quant)
dim(dataFused$quant)
```

getUPS1acc

UniProt Accession-Numbers And Names Of UPS1 Proteins

Description

UPS1 (see <https://www.sigmaaldrich.com/FR/en/product/sigma/ups1>) and UPS2 are commercial products consisting of a mix of 48 human (purified) proteins. They are frequently used as standard in spike-in experiments, available from Sigma-Aldrich (<https://www.sigmaaldrich.com/GB/en>). This function allows accessing their protein accession numbers and associated names on **UniProt**

Usage

```
getUPS1acc(updated = TRUE, silent = FALSE, debug = FALSE, callFrom = NULL)
```

Arguments

updated	(logical) return updated accession number (of UBB)
silent	(logical) suppress messages
debug	(logical) additional messages for debugging
callFrom	(character) allows easier tracking of messages produced

Details

Please note that the UniProt accession 'P62988' for 'UBIQ_HUMAN' (as originally cited by Sigma-Aldrich) has been withdrawn and replaced in 2010 by **UniProt** by the accessions 'P0CG47', 'P0CG48', 'P62979', and 'P62987'. This initial accession is available via `getUPS1acc()$acOld`, now `getUPS1acc()$ac` contains 'P0CG47'.

Value

This function returns data.frame with accession-numbers as stated by the supplier (`$acFull`), trimmed accession-numbers, ie without version numbers (`$ac`), and associated (UniProt) entry-names (`$EntryName`) from **UniProt** as well as the species designation for the collection of 48 human UPS1 or UPS2 proteins.

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

Examples

```
head(getUPS1acc())
```

inspectSpeciesIndic	<i>Inspect Species Indictaion Or Group of Proteins</i>
---------------------	--

Description

This function inspects its main argument to convert a species indication to the scientific name or to return all protein-accession numbers for a name of a standard collection like UPS1.

Usage

```
inspectSpeciesIndic(x, silent = FALSE, debug = FALSE, callFrom = NULL)
```

Arguments

x	(character) species indication or name of collection of proteins (so far only UPS1 & UPS2)
silent	(logical) suppress messages
debug	(logical) display additional messages for debugging
callFrom	(character) allows easier tracking of messages produced

Value

This function returns a character vector

See Also

[getUPS1acc](#);

Examples

```
inspectSpeciesIndic("Human")
inspectSpeciesIndic("UPS1")
```


isolNAneighb

*Isolate NA-Neighbours***Description**

This functions extracts all replicate-values where at least one of the replicates is NA and sorts by number of NAs per group. A list with all NA-neighbours organized by the number of NAs gets returned.

Usage

```
isolNAneighb(mat, gr, silent = FALSE, debug = 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
silent	(logical) suppress messages
debug	(logical) display additional messages for debugging
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,  
  debug = FALSE,  
  callFrom = NULL  
)
```

Arguments

<code>comp</code>	(character) atomic composition
<code>massTy</code>	(character) 'mono' or 'average'
<code>rmEmpty</code>	(logical) suppress empty entries
<code>silent</code>	(logical) suppress messages
<code>debug</code>	(logical) display additional messages for debugging
<code>callFrom</code>	(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**

This function makes histograms of the full data and shows sub-population of NA-neighbour values.

Usage

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

Arguments

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

Details

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 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 measurements in proteomics and transcriptomics this may happen. So this function 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 whether measures containing NA-values should rather be omitted. Of course, such decisions do have a strong impact on further steps of data-analysis and should be performed with care.

Value

This function produces a graphic (to the current graphical device)

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 describes 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 replace all NA-values based on values from a normal distribution resembling their respective NA-neighbours.

Usage

```
matrixNNeighbourImpute(
  dat,
  gr,
  imputMethod = "mode2",
  retnNA = TRUE,
  avSd = c(0.15, 0.5),
  avSdH = NULL,
  NNeigLst = 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

<code>dat</code>	(matrix or data.frame) main data (may contain NA)
<code>gr</code>	(character or factor) grouping of columns of 'dat', replicate association
<code>imputMethod</code>	(character) choose the imputation method (may be 'mode2'(default), 'mode1', 'datQuant', 'modeAdopt' or 'informed')
<code>retnNA</code>	(logical) decide (if =TRUE) only NA-substuted data should be returned, or if list with \$data, \$nNA, \$NNeighbour and \$randParam should be returned
<code>avSd</code>	(numerical,length=2) population characteristics 'high' (mean and sd) for >1 NA-neighbours (per line)
<code>avSdH</code>	deprecated, please use avSd inestad; (numerical,length=2) population characteristics 'high' (mean and sd) for >1 NA-neighbours (per line)
<code>NNeigLst</code>	(list) option for repeated rounds of imputations: list of NA-neighbour values can be furnished for slightly faster processing
<code>plotHist</code>	(character or logical) decide if supplemental figure with histogram should be drawn, the details 'Hist', 'quant' (display quantile of originak data), 'mode' (display mode of original data) can be chosen explicetely
<code>xLab</code>	(character) label on x-axis on plot
<code>xLim</code>	(numeric, length=2) custom x-axis limits
<code>yLab</code>	(character) label on y-axis on plot
<code>yLim</code>	(numeric, length=2) custom y-axis limits
<code>tit</code>	(character) title on plot

<code>figImputDetail</code>	(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)
<code>seedNo</code>	(integer) seed-value for normal random values
<code>silent</code>	(logical) suppress messages
<code>callFrom</code>	(character) allow easier tracking of messages produced
<code>debug</code>	(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 (in particular in 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/proteins may be 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 occurred. 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 `avSd` must be set to NA. Otherwise, the first value `avSd` will be used as quantile of all data to define the mean for the imputed data (ie as `quantile(dat, avSd[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 `avSd` (or `avSd`, 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 described 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

This function plots ROC curves based on results from [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,
  debug = FALSE,
  callFrom = NULL
)

```

Arguments

<code>dat</code>	(matrix) from testing (eg summarizeForROC)
<code>...</code>	optional additional data-sets to include as seprate ROC-curves to same plot (must be of same type of format as 'dat')
<code>useColumn</code>	(integer or character, length=2) columns from dat to be used for pecificity and sensitivity
<code>methNames</code>	(character) names of methods (data-sets) to be displayed
<code>col</code>	(character) custom colors for lines and text (choose one color for each different data-set)
<code>pch</code>	(integer) type of symbol to be used (see also par)
<code>bg</code>	(character) background color in plot (see also par)
<code>tit</code>	(character) custom title
<code>xlim</code>	(numeric, length=2) custom x-axis limits
<code>ylim</code>	(numeric, length=2) custom y-axis limits
<code>point05</code>	(numeric) specific point to highlight in plot (typically at alpha=0.05)
<code>pointSi</code>	(numeric) size of points (as expansion factor cex)
<code>nByMeth</code>	(integer) value of n to display
<code>speciesOrder</code>	(integer) custom order of species in legend
<code>txtLoc</code>	(numeric, length=3) location for text (x, y location and proportional factor for line-offset, default is c(0.4,0.3,0.04))
<code>legCex</code>	(numeric) cex expansion factor for legend (see also par)
<code>las</code>	(numeric) factor for text-orientation (see also par)
<code>addSuplT</code>	(logical) add text with information about precision,accuracy and FDR
<code>silent</code>	(logical) suppress messages
<code>debug</code>	(logical) display additional messages for debugging
<code>callFrom</code>	(character) allows easier tracking of messages produced

Details

This function produces 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](#).

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	<i>Filter Based On Either Number Of total Peptides And Specific Peptides Or Number Of Razor Petides</i>
---------------	---

Description

This function filters based on either a) number of total peptides and specific peptides or b) number of razor petides.

Usage

```
razorNoFilter(
  annot,
  speNa = NULL,
  totNa = NULL,
  minRazNa = NULL,
  minSpeNo = 1,
  minTotNo = 2,
  silent = FALSE,
  debug = 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
debug	(logical) display additional messages for debugging
callFrom	(character) allows easier tracking of messages produced

Details

This function was designed for filtering using a minimum 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).

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")
```

readAlphaPeptFile

Read (Normalized) Quantitation Data Files Produced By AlphaPept

Description

Protein quantification results from **AlphaPept** can be read using this function. Input files compressed as .gz can be read as well. The protein abundance values (XIC) get extracted. Since protein annotation is not very extensive with this format of data, the function allows reading the initial fasta files (from the directory above the quantitation-results) allowing to extract more protein-annotation (like species). Sample-annotation (if available) can be extracted from sdrf files, too. The protein abundance values may be normalized using multiple methods (median normalization as default), the determination of normalization factors can be restricted to specific proteins (normalization to bait protein(s), or to invariable matrix of spike-in experiments). The protein annotation data gets parsed to extract specific fields (ID, name, description, species ...). Besides, a graphical display of the distribution of protein abundance values may be generated before and after normalization.

Usage

```
readAlphaPeptFile(
  fileName = "results_proteins.csv",
  path = NULL,
  fasta = NULL,
  isLog2 = FALSE,
  normalizeMeth = "none",
  quantCol = "_LFQ$",
```

```

    contamCol = NULL,
    read0asNA = TRUE,
    refLi = NULL,
    sampleNames = NULL,
    specPref = NULL,
    extrColNames = NULL,
    remRev = TRUE,
    remConta = FALSE,
    separateAnnot = TRUE,
    gr = NULL,
    sdrf = NULL,
    suplAnnotFile = NULL,
    groupPref = list(lowNumberOfGroups = TRUE, chUnit = TRUE),
    titGraph = NULL,
    wex = 1.6,
    plotGraph = TRUE,
    silent = FALSE,
    debug = FALSE,
    callFrom = NULL
)

```

Arguments

fileName	(character) name of file to be read (default 'results_proteins.csv'). Gz-compressed files can be read, too.
path	(character) path of file to be read
fasta	(logical or character) if TRUE the (first) fasta from one directory higher than fileName will be read as fasta-file to extract further protein annotation; if character a fasta-file at this location will be read/used/
isLog2	(logical) typically data read from AlphaPept are expected NOT to be isLog2=TRUE
normalizeMeth	(character) normalization method, defaults to median, for more 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
read0asNA	(logical) decide if initial quantifications at 0 should be transformed to NA (thus avoid -Inf in log2 results)
refLi	(character or integer) custom specify which line of data should be used for normalization, ie which line 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) custom column-names for quantification data; this argument has priority over suplAnnotFile
specPref	(character) prefix to identifiers allowing to separate i) recognize contamination database, ii) species of main identifications and iii) spike-in species
extrColNames	(character or NULL) custom definition of col-names to extract

remRev	(logical) option to remove all protein-identifications based on reverse-peptides
remConta	(logical) option to remove all proteins identified as contaminants
separateAnnot	(logical) if TRUE output will be organized as list with \$annot, \$abund for initial/raw abundance values and \$quant with final normalized quantitations
gr	(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from sdrf and/or suplAnnotFile (if provided)
sdrf	(character, list or data.frame) optional extraction and adding of experimental meta-data: if character, this may be the ID at ProteomeExchange, the second & third elements may give further indications for automatic organization of groups of replicates. Besides, the output from readSdrf or a list from defineSamples may be provided; if gr is provided, gr gets priority for grouping of replicates; if sdrfOrder=TRUE the output will be put in order of sdrf
suplAnnotFile	(logical or character) optional reading of supplemental files produced by Compomics; if gr is provided, it gets priority for grouping of replicates if TRUE default to files 'summary.txt' (needed to match information of sdrf) and 'parameters.txt' which can be found in the same folder as the main quantitation results; if character the respective file-names (relative or absolute path), 1st is expected to correspond to 'summary.txt' (tabulated text, the samples as given to Compomics) and 2nd to 'parameters.txt' (tabulated text, all parameters given to Compomics)
groupPref	(list) additional parameters for interpreting meta-data to identify structure of groups (replicates), will be passed to readSampleMetaData. May contain lowNumberOfGroups=FALSE for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, ie higher number of samples per group) May contain chUnit (logical or character) to be passed to readSampleMetaData() for (optional) adjustig of unit-prefixes in meta-data group labels, in case multiple different unit-prefixes are used (eg '100pMol' and '1nMol').
titGraph	(character) custom title to plot of distribution of quantitation values
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
debug	(logical) additional messages for debugging
callFrom	(character) allow easier tracking of messages produced

Details

Meta-data describing the samples and experimental setup may be available from a sdrf-file (from the directory above the analysis/quantification results) If available, the meta-data will be examined for determining groups of replicates and the results thereof can be found in \$sampleSetup\$levels. Alternatively, a dataframe formatted like sdrf-files (ie for each sample a separate line, see also function readSdrf) may be given, too.

This import-function has been developed using AlphaPept version x.x. The final output is a list containing these elements: \$raw, \$quant, \$annot, \$counts, \$sampleSetup, \$quantNotes, \$notes, or

(if `separateAnnot=FALSE`) data.frame with annotation- and main quantification-content. If `sdrf` information has been found, an additional list-element `setup` will be added containing the entire meta-data as `setup$meta` and the suggested organization as `setup$lev`.

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', `$quantNotes`, `$notes` and optional `setup` for meta-data from `sdrf`; or a data.frame with quantitation and annotation if `separateAnnot=FALSE`

See Also

[read.table](#), [normalizeThis](#), [readProteomeDiscovererFile](#), [readProlineFile](#) (and other import-functions), [matrixNAinspect](#)

Examples

```
path1 <- system.file("extdata", package="wrProteo")
# Here we'll load a short/trimmed example file
fiNaAP <- "tinyAlpaPeptide.csv.gz"
dataAP <- readAlphaPeptFile(file=fiNaAP, path=path1, tit="tiny AlphaPptide ")
summary(dataAP$quant)
```

readDiaNNFile

Read Tabulated Files Exported By DIA-NN At Protein Level

Description

This function allows importing protein identification and quantification results from **DIA-NN**. Data should be exported as tabulated text (tsv) as protein-groups (pg) to allow import by this function. Quantification data and other relevant information will be parsed and extracted (similar to the other import-functions from this package). The final output is a list containing as (main) elements: `$annot`, `$raw` and `$quant`, or a data.frame with the quantification data and a part of the annotation if argument `separateAnnot=FALSE`.

Usage

```
readDiaNNFile(
  fileName,
  path = NULL,
  normalizeMeth = "median",
  sampleNames = NULL,
  read0asNA = TRUE,
  quantCol = "\\..raw$",
  annotCol = NULL,
  refLi = NULL,
  separateAnnot = TRUE,
```

```

FDRCol = NULL,
groupPref = list(lowNumberOfGroups = TRUE, chUnit = TRUE),
plotGraph = TRUE,
titGraph = "DiaNN",
wex = 1.6,
specPref = c(conta = "CON_|LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
gr = NULL,
sdrf = NULL,
suplAnnotFile = FALSE,
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, defaults to median, for more details see normalizeThis)
sampleNames	(character) custom column-names for quantification data; this argument has priority over suplAnnotFile
read0asNA	(logical) decide if initial quantifications at 0 should be transformed to NA (thus avoid -Inf in log2 results)
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..I", "X..AAs", "MW..kDa."))
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 log2 (normalized) quantitations
FDRCol	- not used (the argument was kept to remain with the same synthax as the other import functions fo this package)
groupPref	(list) additional parameters for interpreting meta-data to identify structure of groups (replicates), will be passed to readSampleMetaData. May contain lowNumberOfGroups=FALSE for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, ie higher number of samples per group) May contain chUnit (logical or character) to be passed to readSampleMetaData() for (optional) adjustig of unit-prefixes in meta-data group labels, in case multiple different unit-prefixes are used (eg '100pMol' and '1nMol').
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
titGraph	(character) custom title to plot of distribution of quantitation values

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 contaminants - will be marked as 'conta', 2nd for main species - marked as 'mainSpe', and optional following ones for supplemental tags/species - marked 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)
gr	(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from sdrf and/or suplAnnotFile (if provided)
sdrf	(character, list or data.frame) optional extraction and adding of experimental meta-data: if character, this may be the ID at ProteomeExchange, the second element may give further indications for automatic organization of groups of replicates. Besides, the output from readSdrf or a list from defineSamples may be provided; if gr is provided, gr gets priority for grouping of replicates
suplAnnotFile	(logical or character) optional reading of supplemental files; however, if gr is provided, gr gets priority for grouping of replicates; if character the respective file-name (relative or absolute path)
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 DIA-NN version 1.8.x. Note, reading gene-group (gg) files is in principle possible, but resulting files typically lack protein-identifiers which may be less convenient in later steps of analysis. Thus, it is suggested to rather read protein-group (pg) files.

Using the argument suplAnnotFile 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 related information.

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](#), [readProtDiscovFile](#), [readProlineFile](#)

Examples

```
diaNNfi1 <- "tinyDiaNN1.tsv.gz"
## This file contains much less identifications than one may usually obtain
path1 <- system.file("extdata", package="wrProteo")
## let's define the main species and allow tagging some contaminants
specPref1 <- c(conta="conta|CON_|LYSC_CHICK", mainSpecies="HUMAN")
dataNN <- readDiaNNFile(path1, file=diaNNfi1, specPref=specPref1, tit="Tiny DIA-NN Data")
summary(dataNN$quant)
```

readDiaNNPeptides	<i>Read Tabulated Files Exported by DiaNN At Peptide Level</i>
-------------------	--

Description

This function allows importing peptide identification and quantification results from **DiaNN**. Data should be exported as tabulated text (tsv) to allow import by this function. Quantification data and other relevant information will be extracted similar like the other import-functions from this package. The final output is a list containing as (main) elements: \$annot, \$raw and \$quant, or a data.frame with the quantification data and a part of the annotation if argument separateAnnot=FALSE.

Usage

```
readDiaNNPeptides(
  fileName,
  path = NULL,
  normalizeMeth = "median",
  sampleNames = NULL,
  read0asNA = TRUE,
  quantCol = "\\..raw$",
  annotCol = NULL,
  refLi = NULL,
  separateAnnot = TRUE,
  FDRCol = NULL,
  groupPref = list(lowNumberOfGroups = TRUE, chUnit = TRUE),
  plotGraph = TRUE,
  titGraph = "DiaNN",
  wex = 1.6,
  specPref = c(conta = "CON_|LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
  gr = NULL,
  sdrf = NULL,
  suplAnnotFile = FALSE,
  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, defaults to median, for more details see normalizeThis)
sampleNames	(character) custom column-names for quantification data; this argument has priority over suplAnnotFile
read0asNA	(logical) decide if initial quantifications at 0 should be transformed to NA (thus avoid -Inf in log2 results)

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..I "X..AAs","MW..kDa."))
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 log2 (normalized) quantitations
FDRCol	(list) - not used
groupPref	(list) additional parameters for interpreting meta-data to identify structure of groups (replicates), will be passed to readSampleMetaData. May contain lowNumberOfGroups=FALSE for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, ie higher number of samples per group) May contain chUnit (logical or character) to be passed to readSampleMetaData() for (optional) adjustig of unit-prefixes in meta-data group labels, in case multiple different unit-prefixes are used (eg '100pMol' and '1nMol').
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
titGraph	(character) custom title to plot of distribution of quantitation values
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 contaminants - will be marked as 'conta', 2nd for main species-marked as 'mainSpe', and optional following ones for supplemental tags/species - marked 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)
gr	(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from sdrf and/or suplAnnotFile (if provided)
sdrf	(character, list or data.frame) optional extraction and adding of experimenal meta-data: if character, this may be the ID at ProteomeExchange, the second element may give further indications for automatic organization of groups of replicates. Besides, the output from readSdrf or a list from defineSamples may be provided; if gr is provided, gr gets priority for grouping of replicates
suplAnnotFile	(logical or character) optional reading of supplemental files; however, if gr is provided, gr gets priority for grouping of replicates; if character the respective file-name (relative or absolute path)
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 DiaNN version 1.8.x.

Using the argument `suplAnnotFile` 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 related information.

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`, `readProtDiscovFile`, `readProlineFile`

Examples

```
diaNNfi1 <- "tinyDiaNN1.tsv.gz"
## This file contains much less identifications than one may usually obtain
path1 <- system.file("extdata", package="wrProteo")
## let's define the main species and allow tagging some contaminants
specPref1 <- c(conta="conta|CON_|LYSC_CHICK", mainSpecies="HUMAN")
dataNN <- readDiaNNFile(path1, file=diaNNfi1, specPref=specPref1, tit="Tiny DIA-NN Data")
summary(dataNN$quant)
```

readFasta2

Read File Of Protein Sequences In Fasta Format

Description

Read fasta formatted file (from **UniProt**) to extract (protein) sequences and name.

Usage

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

Arguments

filename	(character) names fasta-file to be read; .gz compressed files can be read, too (see examples)
delim	(character) delimiter at header-line
databaseSign	(character) characters at beginning right after the '>' (typically specifying the data-base-origin), they will be excluded from the sequence-header
removeEntries	(character) if removeEntries='empty' allows removing entries without any sequence entries; set to removeEntries='duplicated' to remove duplicate entries (same sequence and same header) removeEntries='allNA' remove columns with all entries NA (if tableOut=TRUE)
tableOut	(logical) toggle to return named character-vector or matrix with enhanced parsing of fasta-header. The resulting matrix will contain the columns '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
strictSpecPattern	(logical or character) deprecated, this argument is not used any more
cleanCols	(logical) deprecated, please use argument removeEntries="allNA"
silent	(logical) suppress messages
callFrom	(character) allows easier tracking of messages produced
debug	(logical) supplemental messages for debugging

Details

Read fasta-header -as is (ie without any parsing) : Set argument tableOut=FALSE. If tableOut=TRUE the output will be organized as matrix for separating meta-annotation (eg uniqueIdentifier, entryName, proteinName, GN) in separate columns. Please keep in mind that parsers were primarily designed for the UniProt format.

Value

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

See Also

[writeFasta2](#) for writing as fasta; for reading [readLines](#) or read.fasta from the package [seqinr](#)

Examples

```
## Tiny example with common contaminants
path1 <- system.file('extdata', package='wrProteo')
fiNa <- "conta1.fasta.gz"
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)
```

readFragpipeFile	<i>Read Tabulated Files Exported by FragPipe At Protein Level</i>
------------------	---

Description

This function allows importing protein identification and quantification results from **Fragpipe** which were previously exported as tabulated text (tsv). Quantification data and other relevant information will be extracted similar like the other import-functions from this package. The final output is a list containing the elements: \$annot, \$raw and \$quant, or a data.frame with the quantification data and a part of the annotation if argument separateAnnot=FALSE.

Usage

```
readFragpipeFile(
  fileName,
  path = NULL,
  normalizeMeth = "median",
  sampleNames = NULL,
  read0asNA = TRUE,
  quantCol = "Intensity$",
  annotCol = NULL,
  refLi = NULL,
  separateAnnot = TRUE,
  FDRCol = list("Protein.Probability", lim = 0.99),
  groupPref = list(lowNumberOfGroups = TRUE, chUnit = TRUE),
  plotGraph = TRUE,
  titGraph = "FragPipe",
  wex = 1.6,
  specPref = c(conta = "CON_|LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
  gr = NULL,
  sdrf = NULL,
  suplAnnotFile = FALSE,
  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, defaults to median, for more details see normalizeThis)
sampleNames	(character) custom column-names for quantification data; this argument has priority over suplAnnotFile
read0asNA	(logical) decide if initial quntifications at 0 should be transformed to NA (thus avoid -Inf in log2 results)

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..L","X..AAs","MW..kDa."))
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 log2 (normalized) quantitations
FDRCol	(list) optional indication to search for protein FDR information
groupPref	(list) additional parameters for interpreting meta-data to identify structure of groups (replicates), will be passed to readSampleMetaData. May contain lowNumberOfGroups=FALSE for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, ie higher number of samples per group) May contain chUnit (logical or character) to be passed to readSampleMetaData() for (optional) adjustig of unit-prefixes in meta-data group labels, in case multiple different unit-prefixes are used (eg '100pMol' and '1nMol').
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
titGraph	(character) custom title to plot of distribution of quantitation values
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 contaminants - will be marked as 'conta', 2nd for main species-marked as 'mainSpe', and optional following ones for supplemental tags/species - marked 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)
gr	(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from sdrf and/or suplAnnotFile (if provided)
sdrf	(character, list or data.frame) optional extraction and adding of experimetal meta-data: if character, this may be the ID at ProteomeExchange, the second & third elements may give further indications for automatic organization of groups of replicates. Besides, the output from readSdrf or a list from defineSamples may be provided; if gr is provided, gr gets priority for grouping of replicates; if sdrfOrder=TRUE the output will be put in order of sdrf
suplAnnotFile	(logical or character) optional reading of supplemental files; however, if gr is provided, gr gets priority for grouping of replicates; if character the respective file-name (relative or absolute path)
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 Fragpipe versions 18.0 and 19.0.

Using the argument `suplAnnotFile` 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 related information.

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`, `readProtDiscovFile`, `readProlineFile`

Examples

```
FProfi1 <- "tinyFragpipe1.tsv.gz"
path1 <- system.file("extdata", package="wrProteo")
## let's define the main species and allow tagging some contaminants
specPref1 <- c(conta="conta|CON_|LYSC_CHICK", mainSpecies="MOUSE")
dataFP <- readFragpipeFile(path1, file=FProfi1, specPref=specPref1, tit="Tiny Fragpipe Data")
summary(dataFP$quant)
```

readIonbotPeptides	<i>Read Tabulated Files Exported by Ionbot At Peptide Level</i>
--------------------	---

Description

This function allows importing initial peptide identification and quantification results from **Ionbot** which were exported as tabulated tsv can be imported and relevant information extracted. The final output is a list containing 3 main elements: `$annot`, `$raw` and optional `$quant`, or returns data.frame with entire content of file if `separateAnnot=FALSE`.

Usage

```
readIonbotPeptides(
  fileName,
  path = NULL,
  normalizeMeth = "median",
  sampleNames = NULL,
  gr = NULL,
  sdrf = NULL,
  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,
    suplAnnotFile = TRUE,
    groupPref = list(lowNumberOfGroups = TRUE, chUnit = TRUE),
    titGraph = "Ionbot",
    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, defaults to median, for more details see /link[wrMisc]{normalizeTh
sampleNames	(character) new column-names for quantification data (ProteomeDiscoverer does not automatically use file-names from spectra); this argument has priority over suplAnnotFile
gr	(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from sdrf and/or suplAnnotFile (if provided)
sdrf	(character, list or data.frame) optional extraction and adding of experimental meta-data: if character, this may be the ID at ProteomeExchange, the second & third elements may give further indications for automatic organization of groups of replicates. Besides, the output from readSdrf or a list from defineSamples may be provided; if gr is provided, gr gets priority for grouping of replicates; if sdrfOrder=TRUE the output will be put in order of sdrf
read0asNA	(logical) decide if initial quantifications 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..I", "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 mater on 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
suplAnnotFile	(logical or character) optional reading of supplemental files produced by ProteomeDiscoverer; however, if gr is provided, gr gets priority for grouping of replicates; if TRUE defaults to file '*InputFiles.txt' (needed to match information of sdrf) which can be exported next to main quantitation results; if character the respective file-name (relative or absolute path)
groupPref	(list) additional parameters for interpreting meta-data to identify structure of groups (replicates), will be passed to readSampleMetaData. May contain lowNumberOfGroups=FALSE for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, ie higher number of samples per group) May contain chUnit (logical or character) to be passed to readSampleMetaData() for (optional) adjustig of unit-prefixes in meta-data group labels, in case multiple different unit-prefixes are used (eg '100pMol' and '1nMol').
titGraph	(character) deprecated custom title to plot, please use 'tit'
wex	(integer) relative expansion factor of the violin-plot (will be passed to /link[wrGraph]{vioplotW})
specPref	(character or list) define characteristic text for recognizing (main) groups of species (1st for contaminants - will be marked as 'conta', 2nd for main species-marked as 'mainSpe', and optional following ones for supplemental tags/species - marked 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) allows easier tracking of messages produced

Details

Using the argument suplAnnotFile 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 related information.

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

/link[utils]{read.table}, /link{readMaxQuantFile}, /link{readProteomeDiscovererFile}, /link[wrMisc]{normalizeThis})

Examples

```
path1 <- system.file("extdata", package="wrProteo")
fiIonbot <- "tinyIonbotFile1.tsv.gz"
datIobPep <- readIonbotPeptides(fiIonbot, path=path1)
```

readMassChroQFile	<i>Read tabulated files imported from MassChroQ</i>
-------------------	---

Description

Quantification results using MassChroQ should be initially treated using the R-package MassChroqR (both distributed by the PAPPISO 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,
  titGraph = "MassChroQ",
  wex = NULL,
  specPref = c(conta = "CON_|LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
  gr = NULL,
  sdrf = NULL,
  suplAnnotFile = FALSE,
  groupPref = list(lowNumberOfGroups = TRUE, chUnit = TRUE),
  plotGraph = TRUE,
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
```

Arguments

fileName	(character) name of file to be read (may be tsv, csv, rda or rdata); both US and European csv formats are supported
path	(character) path of file to be read
normalizeMeth	(character) normalization method (will be sent to normalizeThis)
sampleNames	(character) custom column-names for quantification data; this argument has priority over suplAnnotFile

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
titGraph	(character) custom title to plot of distribution of quantitation values
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 contaminants - will be marked as 'conta', 2nd for main species - marked as 'mainSpe', and optional following ones for supplemental tags/species - marked 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)
gr	(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from sdrf and/or suplAnnotFile (if provided)
sdrf	(character, list or data.frame) optional extraction and adding of experimental meta-data: if character, this may be the ID at ProteomeExchange, the second & third elements may give further indications for automatic organization of groups of replicates. Besides, the output from readSdrf or a list from defineSamples may be provided; if gr is provided, gr gets priority for grouping of replicates; if sdrfOrder=TRUE the output will be put in order of sdrf
suplAnnotFile	(logical or character) optional reading of supplemental files produced by ProteomeDiscoverer; however, if gr is provided, gr gets priority for grouping of replicates; if TRUE defaults to file '*InputFiles.txt' (needed to match information of sdrf) which can be exported next to main quantitation results; if character the respective file-name (relative or absolute path)
groupPref	(list) additional parameters for interpreting meta-data to identify structure of groups (replicates), will be passed to readSampleMetaData. May contain lowNumberOfGroups=FALSE for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, ie higher number of samples per group) May contain chUnit (logical or character) to be passed to readSampleMetaData() for (optional) adjustig of unit-prefixes in meta-data group labels, in case multiple different unit-prefixes are used (eg '100pMol' and '1nMol').
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
debug	(logical) additional messages for debugging
callFrom	(character) allow easier tracking of messages produced

Details

The final output of this function 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 PAPPISO (<http://pappso.inrae.fr/>). When saving quantifications generated in R as RData (with extension .rdata or .rda) using the R-packages associated with MassChroq, 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)
```

<code>readMaxQuantFile</code>	<i>Read Quantitation Data-Files (proteinGroups.txt) Produced From MaxQuant At Protein Level</i>
-------------------------------	---

Description

Protein quantification results from **MaxQuant** can be read using this function and relevant information extracted. Input files compressed as .gz can be read as well. The protein abundance values (XIC), peptide counting information like number of unique razor-peptides or PSM values and sample-annotation (if available) can be extracted, too. The protein abundance values may be normalized using multiple methods (median normalization as default), the determination of normalization factors can be restricted to specific proteins (normalization to bait protein(s), or to invariable matrix of spike-in experiments). The protein annotation data gets parsed to extract specific fields (ID, name, description, species ...). Besides, a graphical display of the distribution of protein abundance values may be generated before and after normalization.

Usage

```
readMaxQuantFile(
  path,
  fileName = "proteinGroups.txt",
  normalizeMeth = "median",
  quantCol = "LFQ.intensity",
  contamCol = "Potential.contaminant",
```

```

    pepCountCol = c("Razor + unique peptides", "Unique peptides", "MS.MS.count"),
    read0asNA = TRUE,
    refLi = NULL,
    sampleNames = NULL,
    extrColNames = c("Majority.protein.IDs", "Fasta.headers", "Number.of.proteins"),
    specPref = c(conta = "conta|CON_|LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
    remRev = TRUE,
    remConta = FALSE,
    separateAnnot = TRUE,
    gr = NULL,
    sdrf = NULL,
    suplAnnotFile = NULL,
    groupPref = list(lowNumberOfGroups = TRUE, chUnit = TRUE),
    titGraph = NULL,
    wex = 1.6,
    plotGraph = TRUE,
    silent = FALSE,
    debug = 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, defaults to median, for more 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
pepCountCol	(character) pattern to search among column-names for count data (1st entry for 'Razor + unique peptides', 2nd for 'Unique peptides', 3rd for 'MS.MS.count' (PSM))
read0asNA	(logical) decide if initial quantifications at 0 should be transformed to NA (thus avoid -Inf in log2 results)
refLi	(character or integer) custom specify which line of data should be used for normalization, ie which line 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) custom column-names for quantification data; this argument has priority over suplAnnotFile
extrColNames	(character) column names to be read (1st position: prefix for LFQ quantitation, default 'LFQ.intensity'; 2nd: column name for protein-IDs, default 'Majority.protein.IDs'; 3rd: column names of fasta-headers, default 'Fasta.headers', 4th: 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
remConta	(logical) option to remove all proteins identified as contaminants
separateAnnot	(logical) if TRUE output will be organized as list with \$annot, \$abund for initial/raw abundance values and \$quant with final normalized quantitations
gr	(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from sdrf and/or suplAnnotFile (if provided)
sdrf	(character, list or data.frame) optional extraction and adding of experimental meta-data: if character, this may be the ID at ProteomeExchange, the second & third elements may give further indications for automatic organization of groups of replicates. Besides, the output from readSdrf or a list from defineSamples may be provided; if gr is provided, gr gets priority for grouping of replicates; if sdrfOrder=TRUE the output will be put in order of sdrf
suplAnnotFile	(logical or character) optional reading of supplemental files produced by MaxQuant; if gr is provided, it gets priority for grouping of replicates if TRUE default to files 'summary.txt' (needed to match information of sdrf) and 'parameters.txt' which can be found in the same folder as the main quantitation results; if character the respective file-names (relative or absolute path), 1st is expected to correspond to 'summary.txt' (tabulated text, the samples as given to MaxQuant) and 2nd to 'parameters.txt' (tabulated text, all parameters given to MaxQuant)
groupPref	(list) additional parameters for interpreting meta-data to identify structure of groups (replicates), will be passed to readSampleMetaData. May contain (gr=) vector or factor describing a custom-grouping (will get priority over other sdrf etc) like (gr="sdrf") (global mining of sdrf), (gr="sdrf\$thisColumn") (specific column of sdrf, if not present will fall back to global mining of sdrf), or (gr="colnames") to force grouping based on colnames from main data (after stripping terminal nominators) May contain lowNumberOfGroups=FALSE for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, ie higher number of samples per group) May contain chUnit (logical or character) to be passed to readSampleMetaData() for (optional) adjustig of unit-prefixes in meta-data group labels, in case multiple different unit-prefixes are used (eg '100pMol' and '1nMol').
titGraph	(character) custom title to plot of distribution of quantitation values
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
debug	(logical) additional messages for debugging
callFrom	(character) allow easier tracking of messages produced

Details

MaxQuant is proteomics quantification software provided by the MaxPlanck institute. By default MaxQuant writes the results of each run to the path combined/txt, from there (only) the files 'proteinGroups.txt' (main quantitation at protein level), 'summary.txt' and 'parameters.txt' will be used.

Meta-data describing the samples and experimental setup may be available from two sources : a) The file summary.txt which gets produced by MaxQuant in the same folder as the main quantification data. b) Furthermore, meta-data deposited as sdrf at Pride can be retrieved (via the respective github page) when giving the accession number in argument sdrf. Then, the meta-data will be examined for determining groups of replicates and the results thereof can be found in \$sampleSetup\$levels. Alternatively, a dataframe formatted like sdrf-files (ie for each sample a separate line, see also function readSdrf) may be given. In tricky cases it is also possible to precise the column-name to use for defining the groups of replicates or the method for automatically choosing the most suited column via the 2nd value of the argument sdrf. Please note, that sdrf is still experimental and only a small fraction of proteomics-data on Pride have been annotated accordingly. If a valid sdrf is furnished, it's information has priority over the information extracted from the MaxQuant produced file summary.txt.

This import-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, \$sampleSetup, \$quantNotes, \$notes, or (if separateAnnot=FALSE) data.frame with annotation- and main quantification-content. If sdrf information has been found, an additional list-element setup will be added containing the entire meta-data as setup\$meta and the suggested organization as setup\$lev.

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, \$notes and optional setup for meta-data from sdrf; or a data.frame with quantitation and annotation if separateAnnot=FALSE

See Also

[read.table](#), [normalizeThis](#), [readProteomeDiscovererFile](#), [readProlineFile](#) (and other imprtfunctions), [matrixNAinspect](#)

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))
```

readMaxQuantPeptides	<i>Read Peptide Identification And Quantitation Data-Files (peptides.txt) Produced By MaxQuant</i>
----------------------	--

Description

Peptide level identification and quantification data produced by **MaxQuant** can be read using this function and relevant information extracted. Input files compressed as .gz can be read as well. The peptide abundance values (XIC), peptide counting information and sample-annotation (if available) can be extracted, too.

Usage

```
readMaxQuantPeptides(
  path,
  fileName = "peptides.txt",
  normalizeMeth = "median",
  quantCol = "Intensity",
  contamCol = "Potential.contaminant",
  pepCountCol = "Experiment",
  refLi = NULL,
  sampleNames = NULL,
  extrColNames = c("Sequence", "Proteins", "Leading.razor.protein", "Start.position",
    "End.position", "Mass", "Missed.cleavages", "Unique..Groups.", "Unique..Proteins.",
    "Charges"),
  specPref = c(conta = "conta|CON_|LYSC_CHICK", mainSpecies = "HUMAN"),
  remRev = TRUE,
  remConta = FALSE,
  separateAnnot = TRUE,
  gr = NULL,
  sdrf = NULL,
  suplAnnotFile = NULL,
  groupPref = list(lowNumberOfGroups = TRUE, chUnit = TRUE),
  titGraph = NULL,
  wex = 1.6,
  plotGraph = TRUE,
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
```

Arguments

path	(character) path of file to be read
fileName	(character) name of file to be read (default 'peptides.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
pepCountCol	(character) pattern to search among column-names for count data (defaults to 'Experiment')
refLi	(character or integer) custom specify which line of data should be used for normalization, ie which line 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) custom column-names for quantification data; this argument has priority over suplAnnotFile
extrColNames	(character) column names to be read (1st position: prefix for quantitation, default 'intensity'; 2nd: column name for peptide-IDs, default)
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 peptide-identifications based on reverse-peptides
remConta	(logical) option to remove all peptides identified as contaminants
separateAnnot	(logical) if TRUE output will be organized as list with \$annot, \$abund for initial/raw abundance values and \$quant with final normalized quantitations
gr	(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from sdrf and/or suplAnnotFile (if provided)
sdrf	(character, list or data.frame) optional extraction and adding of experimental meta-data: if character, this may be the ID at ProteomeExchange, the second & third elements may give further indications for automatic organization of groups of replicates. Besides, the output from readSdrf or a list from defineSamples may be provided; if gr is provided, gr gets priority for grouping of replicates; if sdrfOrder=TRUE the output will be put in order of sdrf
suplAnnotFile	(logical or character) optional reading of supplemental files produced by MaxQuant; if gr is provided, it gets priority for grouping of replicates if TRUE default to files 'summary.txt' (needed to match information of sdrf) and 'parameters.txt' which can be found in the same folder as the main quantitation results; if character the respective file-names (relative or absolute path), 1st is expected to correspond to 'summary.txt' (tabulated text, the samples as given to MaxQuant) and 2nd to 'parameters.txt' (tabulated text, all parameters given to MaxQuant)
groupPref	(list) additional parameters for interpreting meta-data to identify structure of groups (replicates), will be passed to readSampleMetaData. May contain lowNumberOfGroups=FALSE for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, ie higher number of samples per group) May contain chUnit (logical or character) to be passed to readSampleMetaData() for (optional) adjustig of unit-prefixes in meta-data group labels, in case multiple different unit-prefixes are used (eg '100pMol' and '1nMol').
titGraph	(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
debug	(logical) additional messages for debugging
callFrom	(character) allows easier tracking of messages produced

Details

The peptide annotation data gets parsed to extract specific fields (ID, name, description, species ...). Besides, a graphical display of the distribution of peptide abundance values may be generated before and after normalization.

MaxQuant is proteomics quantification software provided by the MaxPlanck institute. By default MaxQuant write the results of each run to the path combined/txt, from there (only) the files 'peptides.txt' (main quantitation at peptide level), 'summary.txt' and 'parameters.txt' will be used for this function.

Meta-data describing the samples and experimental setup may be available from two sources : a) The file summary.txt which gets produced by MaxQuant in the same folder as the main quantification data. b) Furthermore, meta-data deposited as sdrf at Pride can be retrieved (via the respective github page) when giving the accession number in argument sdrf. Then, the meta-data will be examined for determining groups of replicates and the results thereof can be found in \$sampleSetup\$levels. Alternatively, a dataframe formatted like sdrf-files (ie for each sample a separate line, see also function readSdrf) may be given. In tricky cases it is also possible to precise the column-name to use for defining the groups of replicates or the method for automatically choosing the most suited column via the 2nd value of the argument sdrf, see also the function defineSamples (which gets used internally). Please note, that sdrf is still experimental and only a small fraction of proteomics-data on Pride have been annotated accordingly. If a valid sdrf is furnished, it's information has priority over the information extracted from the MaxQuant produced file summary.txt.

This function has been developed using MaxQuant versions 1.6.10.x to 2.0.x, the format of the resulting file 'peptides.txt' is typically well conserved between versions. The final output is a list containing these elements: \$raw, \$quant, \$annot, \$counts, \$sampleSetup, \$quantNotes, \$notes, or (if separateAnnot=FALSE) data.frame with annotation- and main quantification-content. If sdrf information has been found, an additional list-element setup will be added containing the entire meta-data as setup\$meta and the suggested organization as setup\$lev.

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, \$notes and optional setup for meta-data from sdrf; or a data.frame with quantitation and annotation if separateAnnot=FALSE

See Also

[read.table](#), [normalizeThis](#)), for reading protein level [readMaxQuantFile](#), [readProlineFile](#)

Examples

```
# Here we'll load a short/trimmed example file (thus not the MaxQuant default name)
MQpepFi1 <- "peptides_tinyMQ.txt.gz"
path1 <- system.file("extdata", package="wrProteo")
specPref1 <- c(conta="conta|CON_|LYSC_CHICK", mainSpecies="YEAST", spec2="HUMAN")
dataMQpep <- readMaxQuantPeptides(path1, file=MQpepFi1, specPref=specPref1,
  tit="Tiny MaxQuant Peptides")
summary(dataMQpep$quant)
```

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 normaiized 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) minimum number of peptides to be used for rerunning 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 contaminants - will be marked as 'conta', 2nd for main species - marked as 'mainSpe', and optional following ones for supplemental tags/species - marked 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 compressed 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	<i>Read Xlsx, Csv Or Tsv Files Exported From Proline And MS-Angel</i>
-----------------	---

Description

Quantification results from Proline **Proline** and MS-Angel exported as xlsx format can be read directly using this function. Besides, files in tsv, csv (European and US format) or tabulated txt can be read and interpreted, too.

Usage

```
readProlineFile(
  fileName,
  path = NULL,
  normalizeMeth = "median",
  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,
  titGraph = NULL,
  wex = 2,
  specPref = c(conta = "_conta\\|", mainSpecies = "OS=Homo sapiens"),
  gr = NULL,
  sdrf = NULL,
  suplAnnotFile = TRUE,
  groupPref = list(lowNumberOfGroups = TRUE, chUnit = TRUE),
  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 should 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) custom column-names for quantification data; this argument has priority over suplAnnotFile
quantCol	(character or integer) columns 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
titGraph	(character) custom title to plot of distribution of quantitation values
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 contaminants - will be marked as 'conta', 2nd for main species-marked as 'mainSpe', and optional following ones for supplemental tags/species - marked 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)
gr	(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from sdrf and/or suplAnnotFile (if provided)
sdrf	(character, list or data.frame) optional extraction and adding of experimental meta-data: if character, this may be the ID at ProteomeExchange, the second & third elements may give further indications for automatic organization of groups of replicates. Besides, the output from readSdrf or a list from defineSamples may be provided; if gr is provided, gr gets priority for grouping of replicates; if sdrfOrder=TRUE the output will be put in order of sdrf

suplAnnotFile	(logical or character) optional reading of supplemental files produced by quantification software; however, if gr is provided, gr gets priority for grouping of replicates; if TRUE defaults to file '*InputFiles.txt' (needed to match information of sdrf) which can be exported next to main quantitation results; if character the respective file-name (relative or absolute path)
groupPref	(list) additional parameters for interpreting meta-data to identify structure of groups (replicates), will be passed to readSampleMetaData. May contain lowNumberOfGroups=FALSE for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, ie higher number of samples per group) May contain chUnit (logical or character) to be passed to readSampleMetaData() for (optional) adjustig of unit-prefixes in meta-data group labels, in case multiple different unit-prefixes are used (eg '100pMol' and '1nMol').
silent	(logical) suppress messages
callFrom	(character) allow easier tracking of messages produced
debug	(logical) display additional messages for debugging

Details

After reading the specified quantification-file 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 (distributions should align).

This function has been developed using Proline version 1.6.1 coupled with MS-Angel 1.6.1. The classical way of using this 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 this 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(path=path1, file=fiNa)
summary(dataABC$quant)
```

```
readProtDiscovererPeptides
      readProtDiscovererPeptides, deprecated
```

Description

This function has been deprecated and replaced by [readProteomeDiscovererPeptides](#) (from this package).

Usage

```
readProtDiscovererPeptides(...)
```

Arguments

... Actually, this function doesn't ready any input any more

Value

This function returns NULL

See Also

[readProteomeDiscovererFile](#), [readProteomeDiscovererPeptides](#)

```
readProtDiscovFile      Read Tabulated Files Exported By ProteomeDiscoverer At Protein
                        Level, Deprecated
```

Description

Deprecated old version of Protein identification and quantification results from **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. Please use readProteomeDiscovererFile() from the same package instead !

Usage

```
readProtDiscovFile(
  fileName,
  path = NULL,
  normalizeMeth = "median",
  sampleNames = NULL,
  read0asNA = TRUE,
  quantCol = "^Abundances*",
```

```

annotCol = NULL,
contamCol = "Contaminant",
refLi = NULL,
separateAnnot = TRUE,
FDRCol = list(c("^Protein.FDR.Confidence", "High"), c("^Found.in.Sample.", "High")),
gr = NULL,
sdrf = NULL,
suplAnnotFile = TRUE,
groupPref = list(lowNumberOfGroups = TRUE),
specPref = c(conta = "CON_|LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
plotGraph = TRUE,
wex = 1.6,
titGraph = "Proteome Discoverer",
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, defaults to median, for more details see normalizeThis)
sampleNames	(character) custom column-names for quantification data (ProteomeDiscoverer does not automatically use file-names from spectra); this argument has priority over suplAnnotFile
read0asNA	(logical) decide if initial quantifications 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..I","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 mater on 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 log2 (normalized) quantitations
FDRCol	(list) optional indication to search for protein FDR information
gr	(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from sdrf and/or suplAnnotFile (if provided)

sdrf	(character, list or data.frame) optional extraction and adding of experimental meta-data: if character, this may be the ID at ProteomeExchange, the second element may give further indications for automatic organization of groups of replicates. Besides, the output from readSdrf or a list from defineSamples may be provided; if gr is provided, gr gets priority for grouping of replicates
suplAnnotFile	(logical or character) optional reading of supplemental files produced by ProteomeDiscoverer; however, if gr is provided, gr gets priority for grouping of replicates; if TRUE defaults to file '*InputFiles.txt' (needed to match information of sdrf) which can be exported next to main quantitation results; if character the respective file-name (relative or absolute path)
groupPref	(list) additional parameters for interpreting meta-data to identify structure of groups (replicates), will be passed to readSampleMetaData. May contain lowNumberOfGroups=FALSE for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, ie higher number of samples per group)
specPref	(character or list) define characteristic text for recognizing (main) groups of species (1st for contaminants - will be marked as 'conta', 2nd for main species-marked as 'mainSpe', and optional following ones for supplemental tags/species - marked 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 or integer) optional plot of type vioplot of initial and normalized data (using normalizeMeth); if integer, it will be passed to layout when plotting
wex	(integer) relative expansion factor of the violin-plot (will be passed to vioplotW)
titGraph	(character) custom title to plot of distribution of quantitation values
silent	(logical) suppress messages
debug	(logical) additional messages for debugging
callFrom	(character) allow easier tracking of messages produced

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](#), [readFragpipeFile](#)

Examples

```
path1 <- system.file("extdata", package="wrProteo")
fiNa <- "tinyPD_allProteins.txt.gz"
## Please use the function readProteinDiscovererFile(), as shown below (same syntax)
dataPD <- readProteomeDiscovererFile(file=fiNa, path=path1, suplAnnotFile=FALSE)
summary(dataPD$quant)
```

readProteomeDiscovererFile

Read Tabulated Files Exported By ProteomeDiscoverer At Protein Level

Description

This function allows importing protein identification and quantification results from **Thermo ProteomeDiscoverer** which were exported as tabulated text can be imported and relevant information extracted.

Usage

```
readProteomeDiscovererFile(
  fileName,
  path = NULL,
  normalizeMeth = "median",
  sampleNames = NULL,
  read0asNA = TRUE,
  quantCol = "^Abundance",
  annotCol = NULL,
  contamCol = "Contaminant",
  refLi = NULL,
  separateAnnot = TRUE,
  FDRCol = list(c("^Protein.FDR.Confidence", "High"), c("^Found.in.Sample.", "High")),
  gr = NULL,
  sdrf = NULL,
  suplAnnotFile = TRUE,
  groupPref = list(lowNumberOfGroups = TRUE, chUnit = TRUE),
  specPref = c(conta = "CON_|LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
  plotGraph = TRUE,
  wex = 1.6,
  titGraph = "Proteome Discoverer",
  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, defaults to median, for more details see normalizeThis)
sampleNames	(character) custom column-names for quantification data (ProteomeDiscoverer does not automatically use file-names from spectra); this argument has priority over suplAnnotFile

read0asNA	(logical) decide if initial quantifications at 0 should be transformed to NA
quantCol	(character or integer) define ywhich columns should be extracted as quantitation data : The argument may be the exact column-names to be used, or if length=1 content of quantCol will be used as pattern to search among column-names for \$quant using grep; if quantCol='allAfter_calc.pI' all columns to the right of the column 'calc.pI' will be interpreted as quantitation data (may be useful with files that have been manually edited before passing to wrProteo)
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..I "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 mater on 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 log2 (normalized) quantitations
FDRCol	(list) optional indication to search for protein FDR information
gr	(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from sdrf and/or suplAnnotFile (if provided)
sdrf	(character, list or data.frame) optional extraction and adding of experimetal meta-data: if character, this may be the ID at ProteomeExchange, the second & third elements may give further indications for automatic organization of groups of replicates. Besides, the output from readSdrf or a list from defineSamples may be provided; if gr is provided, gr gets priority for grouping of replicates; if sdrfOrder=TRUE the output will be put in order of sdrf
suplAnnotFile	(logical or character) optional reading of supplemental files produced by ProteomeDiscoverer; however, if gr is provided, gr gets priority for grouping of replicates; if TRUE defaults to file '*InputFiles.txt' (needed to match information of sdrf) which can be exported next to main quantitation results; if character the respective file-name (relative or absolute path)
groupPref	(list) additional parameters for interpreting meta-data to identify structure of groups (replicates), will be passed to readSampleMetaData. May contain lowNumberOfGroups=FALSE for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, ie higher number of samples per group) May contain chUnit (logical or character) to be passed to readSampleMetaData() for (optional) adjustig of unit-prefixes in meta-data group labels, in case multiple different unit-prefixes are used (eg '100pMol' and '1nMol').
specPref	(character or list) define characteristic text for recognizing (main) groups of species (1st for contaminants - will be marked as 'conta', 2nd for main species-marked as 'mainSpe', and optional following ones for supplemental tags/species - marked 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 or integer) optional plot of type vioplot of initial and normalized data (using normalizeMeth); if integer, it will be passed to layout when plotting
wex	(integer) relative expansion factor of the violin-plot (will be passed to vioplotW)
titGraph	(character) custom title to plot of distribution of quantitation values
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 `suplAnnotFile` 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 related information. If a column named `contamCol` is found, the data will be mater on filtered to remove all contaminants, set to NULL for keeping all contaminants.

The final output is a list containing as (main) elements: `$annot`, `$raw` and optional `$quant`, or returns `data.frame` with entire content of file if `separateAnnot=FALSE`.

This function replaces the deprecated function `readProtDiscovFile` which will soon be retracted from this package.

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](#), [readFragpipeFile](#)

Examples

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

readProteomeDiscovererPeptides

Read Tabulated Files Exported By ProteomeDiscoverer At Peptide Level

Description

Initials petide identificationa and 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

```
readProteomeDiscovererPeptides(
  fileName,
  path = NULL,
  normalizeMeth = "median",
  sampleNames = NULL,
  suplAnnotFile = TRUE,
  gr = NULL,
  sdrf = NULL,
  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,
  titGraph = "Proteome Discoverer",
  wex = 1.6,
  specPref = c(conta = "CON_|LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
  groupPref = list(lowNumberOfGroups = TRUE, chUnit = TRUE),
  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, defaults to median, for more details see normalizeThis)
sampleNames	(character) new column-names for quantification data (ProteomeDiscoverer does not automatically use file-names from spectra); this argument has priority over suplAnnotFile

suplAnnotFile	(logical or character) optional reading of supplemental files produced by ProteomeDiscoverer; however, if gr is provided, gr gets priority for grouping of replicates; if TRUE defaults to file '*InputFiles.txt' (needed to match information of sdrf) which can be exported next to main quantitation results; if character the respective file-name (relative or absolute path)
gr	(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from sdrf and/or suplAnnotFile (if provided)
sdrf	(character, list or data.frame) optional extraction and adding of experimental meta-data: if character, this may be the ID at ProteomeExchange, the second element may give further indications for automatic organization of groups of replicates. Besides, the output from readSdrf or a list from defineSamples may be provided; if gr is provided, gr gets priority for grouping of replicates
read0asNA	(logical) decide if initial quantifications 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..I", "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 mater on 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
titGraph	(character) deprecated 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 contaminants - will be marked as 'conta', 2nd for main species-marked as 'mainSpe', and optional following ones for supplemental tags/species - marked 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)
groupPref	(list) additional parameters for interpreting meta-data to identify structure of groups (replicates), will be passed to readSampleMetaData. May contain lowNumberOfGroups=FALSE for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, ie higher number of samples per group) May contain chUnit (logical or character) to be passed to readSampleMetaData() for (optional) adjustig of unit-prefixes in meta-data group labels, in case multiple different unit-prefixes are used (eg '100pMol' and '1nMol').

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 `suplAnnotFile` 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 related information. Precedent and following aminoacids (relative to identified protease recognition sites) will be removed from peptide sequences and be displayed in `$annot` as columns 'prec' and 'foll'. If a column named `contamCol` is found, the data will be mater on filtered to remove all contaminants, set to NULL for keeping all contaminants This function replaces the deprecated function `readPDEExport`.

Besides, ProteomeDiscoverer version number and full raw-file path will be extracted for `$notes` in final output.

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](#), [readProteomeDiscovererFile](#)

Examples

```
path1 <- system.file("extdata", package="wrProteo")
```

readSampleMetaData	<i>Read Sample Meta-data From Quantification-Software And/Or Sdrf And Align To Experimental Data</i>
--------------------	--

Description

Sample/experimental annotation meta-data from **MaxQuant**, ProteomeDiscoverer, FragPipe, Proline or similar, can be read using this function and relevant information extracted. Furthermore, annotation in **sdrf-format** can be added (the order of sdrf will be adjated automatically, if possible). This functions returns a list with grouping of samples into replicates and additional information gathered. Input files compressed as .gz can be read as well.

Usage

```
readSampleMetaData(
  quantMeth,
  sdrf = NULL,
  suplAnnotFile = NULL,
  path = ".",
  abund = NULL,
  groupPref = list(lowNumberOfGroups = TRUE, sampleNames = NULL, gr = NULL),
  chUnit = TRUE,
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
```

Arguments

quantMeth	(character, length=1) quantification method used; 2-letter abbreviations like 'MQ', 'PD', 'PL', 'FP' etc may be used
sdrf	(character, list or data.frame) optional extraction and adding of experimental meta-data: This may be a matrix or data.frame with information respective to the experimental setup (to understand which lines=samples should be evaluated as replicates). If <code>_sdrf</code> is a character vector, the first entry will be interpreted as path to a local file or the ID/name at ProteomeExchange. Here it is possible to indicate if specific columns of the sdrf should be skipped (<code>_sdrf=c(myFile, skipCol="abc")</code>) For mining the experimental setup/structure sdrf will get priority over suplAnnotFile.
suplAnnotFile	(logical or character) optional reading of supplemental files produced by MaxQuant; if <code>gr</code> is provided, it gets priority for grouping of replicates if TRUE in case of <code>method=='MQ'</code> (MaxQuant) default to files 'summary.txt' (needed to match information of sdrf) and 'parameters.txt' which can be found in the same folder as the main quantitation results; if character the respective file-names (relative or absolute path), 1st is expected to correspond to 'summary.txt' (tabulated text, the samples as given to MaxQuant) and 2nd to 'parameters.txt' (tabulated text, all parameters given to MaxQuant) in case of <code>method=='PL'</code> (Proline), this argument should contain the initial file-name (for the identification and quantification data) in the first position
path	(character) optional path of file(s) to be read
abund	(matrix or data.frame) experimental quantitation data; only column-names will be used for aligning order of annotated samples
groupPref	(list) additional parameters for interpreting meta-data to identify structure of groups (replicates); May contain <code>lowNumberOfGroups=FALSE</code> for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, ie higher number of samples per group). A vector of custom sample-names may be provided via <code>sampleNames=...</code> (must be of correct length); if contains <code>sampleNames="sdrf"</code> sample-names will be used from trimmed file-names out of sdrf.

chUnit	(logical or character) optional adjustig of group-labels from sample meta-data in case multipl different unit-prefixes are used to single common prefix (eg adjust '100pMol' and '1nMol' to '100pMol' and '1000pMol') for better downstream analysis. This option will call adjustUnitPrefix and checkUnitPrefix from package wrMisc If character exatecly this/these unit-names will be searched in sample-names and checked if multiple different decimal prefixes are used; if TRUE the default set of unit-names ('Mol','mol', 'days','day','m','sec','s','h') will be checked in the sample-names for different decimal prefixes
silent	(logical) suppress messages if TRUE
debug	(logical) additional messages for debugging
callFrom	(character) allows easier tracking of messages produced

Details

When initially reading/importing quantitation data, typically very little is known about the setup of different samples in the underlying experiment. The overall aim is to read and mine the corresponding sample-annotation documeneted by the quantitation-software and/or from n sdrf repository and to attach it to the experimental data. This way, in subsequent steps of analysis (eg PCA, statistical tests) the user does not have to bother stuying the experimental setup to figure out which samples should be considered as relicate of whom.

Sample annotation meta-data can be obtained from two sources : a) form additional files produced (and exported) by the initial quantitation software (so far MaxQuant and ProteomeDiscoverer have een implemeneted) or b) from the universal sdrf-format (from Pride or user-supplied). Both types can be imported and checked in the same run, if valid sdrf-information is found this will be given priority. For more information about the sdrf format please see [sdrf on github](#).

Value

This function returns a list with \$groups and \$level (grouping of samples given as integer), and \$meth (method by which grouping as determined). If valid sdrf was given, the resultant list contains in addition \$sdrfDat (data.frame of annotation). Alternatively it may contain a \$sdrfExport if sufficient information has been gathered (so far only for MaxQuant) for a draft sdrf for export (that should be revised and completed by the user). If software annotation has been found it will be shown in \$annotBySoft. If all entries are invalid or entries do not pass the tests, this functions returns an empty list.

See Also

This function is used internally by [readMaxQuantFile](#),/link{readProteomeDiscovererFile} etc; uses [readSdrf](#) for reading sdrf-files, [replicateStructure](#) for mining annotation columns

Examples

```
sdrf001819Setup <- readSampleMetaData(quantMeth=NA, sdrf="PXD001819")
str(sdrf001819Setup)
```

readSdrf

*Read Proteomics Meta-Data As/From Sdrf File***Description**

This function allows reading proteomics meta-data from sdrf file, as they are provided on <https://github.com/bigbio/sdrf-annotated-datasets>. A data.frame containing all annotation data will be returned. To stay conform with the (non-obligatory) recommendations, columnnames are shown as lower caps.

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 messages produced
debug	(logical) display additional messages for debugging

Details

The packages utils and wrMisc must be installed. Please note that reading sdrf files (if not provided as local copy) will take a few seconds, depending on the responsiveness of github. This function only handles the main reading of sdrf data and some diagnostic checks. For mining sdrf data please look at [replicateStructure](#) and [readSampleMetaData](#).

The migration of data within github in April 2026 has been implemented as documented in the bigbio project.

Value

This function returns the content of sdrf-file as data.frame (or NULL if the corresponding file was not found)

See Also

[readSampleMetaData](#), [replicateStructure](#),

Examples

```
## This may take a few sconds...
sdrf001819 <- readSdrf("PXD001819")
str(sdrf001819)
```

readUCSCtable	<i>Read Annotation Files From UCSC</i>
---------------	--

Description

This function allows reading and importing genomic **UCSC-annotation** data. Files can be read as default UCSC exprot or as GTF-format.

Usage

```
readUCSCtable(
  fiName,
  exportFileNa = NULL,
  gtf = NA,
  simplifyCols = c("gene_id", "chr", "start", "end", "strand", "frame"),
  silent = FALSE,
  debug = 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
debug	(logical) display additional messages for debugging
callFrom	(character) allow easier tracking of message(s) produced

Details

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 exported as file for an additional step 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 annotation based on an initial Fasta-file.

Value

This function returns a matrix, optionally the file 'exportFileName' 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, exportFileName=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	<i>Read Protein Annotation As Exported From UniProt Batch-Conversion</i>
-------------------	--

Description

This function allows reading and importing protein-ID conversion results from [UniProt](#).

Usage

```
readUniProtExport(
  UniProtFileName,
  deUcsc = NULL,
  targRegion = NULL,
  useUniPrCol = NULL,
  silent = FALSE,
```

```

    debug = FALSE,
    callFrom = NULL
  )

```

Arguments

UniProtFileNa	(character) name (and path) of file exported from Uniprot (tabulated text file including 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 columns from UniProt exported file should be used/imported (default 'EnsID','Entry','Entry.name','Status','Protein.names','Gene.names','Length').
silent	(logical) suppress messages
debug	(logical) display additional messages for debugging
callFrom	(character) allow easier tracking of message(s) produced

Details

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 retrieve 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 (Ensembl accessions and chromosomal locations, obtained using readUCSCTable) can be integrated.

In a typical use case, first chromosomal 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 Ensembl transcript IDs which can be converted to 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, \$Ensembl, \$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)
```

readWombatNormFile	<i>Read (Normalized) Quantitation Data Files Produced By Wombat At Protein Level</i>
--------------------	--

Description

Protein quantification results from **Wombat-P** using the Bioconductor package Normalizer can be read using this function and relevant information extracted. Input files compressed as .gz can be read as well. The protein abundance values (XIC), peptide counting get extracted. Since protein annotation is not very extensive with this format of data, the function allows reading the initial fasta files (from the directory above the quantitation-results) allowing to extract more protein-annotation (like species). Sample-annotation (if available) can be extracted from sdrf files, which are typically part of the Wombat output, too. The protein abundance values may be normalized using multiple methods (median normalization as default), the determination of normalization factors can be restricted to specific proteins (normalization to bait protein(s), or to invariable matrix of spike-in experiments). The protein annotation data gets parsed to extract specific fields (ID, name, description, species ...). Besides, a graphical display of the distribution of protein abundance values may be generated before and after normalization.

Usage

```
readWombatNormFile(
  fileName,
  path = NULL,
  quantSoft = "(quant software not specified)",
  fasta = NULL,
  isLog2 = TRUE,
  normalizeMeth = "none",
  quantCol = "abundance_",
  contamCol = NULL,
  pepCountCol = c("number_of_peptides"),
```

```

read0asNA = TRUE,
refLi = NULL,
sampleNames = NULL,
extrColNames = c("protein_group"),
specPref = NULL,
remRev = TRUE,
remConta = FALSE,
separateAnnot = TRUE,
gr = NULL,
sdrf = NULL,
suplAnnotFile = NULL,
groupPref = list(lowNumberOfGroups = TRUE, chUnit = TRUE),
titGraph = NULL,
wex = 1.6,
plotGraph = TRUE,
silent = FALSE,
debug = FALSE,
callFrom = NULL
)

```

Arguments

fileName	(character) name of file to be read (default 'proteinGroups.txt' as typically generated by Compomics in txt folder). Gz-compressed files can be read, too.
path	(character) path of file to be read
quantSoft	(character) quantification-software used inside Wombat-P
fasta	(logical or character) if TRUE the (first) fasta from one directory higher than fileName will be read as fasta-file to extract further protein annotation; if character a fasta-file at this location will be read/used/
isLog2	(logical) typically data read from Wombat are expected to be isLog2=TRUE
normalizeMeth	(character) normalization method, defaults to median, for more 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
pepCountCol	(character) pattern to search among column-names for count data (1st entry for 'Razor + unique peptides', 2nd for 'Unique peptides', 3rd for 'MS.MS.count' (PSM))
read0asNA	(logical) decide if initial quantifications at 0 should be transformed to NA (thus avoid -Inf in log2 results)
refLi	(character or integer) custom specify which line of data should be used for normalization, ie which line 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) custom column-names for quantification data; this argument has priority over suplAnnotFile

extrColNames	(character) column names to be read (1st position: prefix for LFQ quantitation, default 'LFQ.intensity'; 2nd: column name for protein-IDs, default 'Majority.protein.IDs'; 3rd: column names of fasta-headers, default 'Fasta.headers', 4th: 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
remConta	(logical) option to remove all proteins identified as contaminants
separateAnnot	(logical) if TRUE output will be organized as list with \$annot, \$abund for initial/raw abundance values and \$quant with final normalized quantitations
gr	(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from sdrf and/or suplAnnotFile (if provided)
sdrf	(logical, character, list or data.frame) optional extraction and adding of experimental meta-data: if sdrf=TRUE the 1st sdrf in the directory above fileName will be used if character, this may be the ID at ProteomeExchange, the second element may give further indications for automatic organization of groups of replicates. Besides, the output from readSdrf or a list from defineSamples may be provided; if gr is provided, gr gets priority for grouping of replicates
suplAnnotFile	(logical or character) optional reading of supplemental files produced by Compomics; if gr is provided, it gets priority for grouping of replicates if TRUE default to files 'summary.txt' (needed to match information of sdrf) and 'parameters.txt' which can be found in the same folder as the main quantitation results; if character the respective file-names (relative or absolute path), 1st is expected to correspond to 'summary.txt' (tabulated text, the samples as given to Compomics) and 2nd to 'parameters.txt' (tabulated text, all parameters given to Compomics)
groupPref	(list) additional parameters for interpreting meta-data to identify structure of groups (replicates), will be passed to readSampleMetaData. May contain lowNumberOfGroups=FALSE for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, ie higher number of samples per group) May contain chUnit (logical or character) to be passed to readSampleMetaData() for (optional) adjustig of unit-prefixes in meta-data group labels, in case multiple different unit-prefixes are used (eg '100pMol' and '1nMol').
titGraph	(character) custom title to plot of distribution of quantitation values
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
debug	(logical) additional messages for debugging
callFrom	(character) allow easier tracking of messages produced

Details

By standard workflow of Wombat-P writes the results of each analysis-method/quantification-algorithm as .csv files Meta-data describing the proteins may be available from two sources : a) The 1st column of the Wombat/normalizer output. b) Form the .fasta file in the directory above the analysis/quantification results of the Wombar-workflow

Meta-data describing the samples and experimental setup may be available from a sdrf-file (from the directory above the analysis/quantification results) If available, the meta-data will be examined for determining groups of replicates and the results thereof can be found in \$sampleSetup\$levels. Alternatively, a dataframe formatted like sdrf-files (ie for each sample a separate line, see also function readSdrf) may be given, too.

This import-function has been developed using Wombat-P version 1.x. The final output is a list containing these elements: \$raw, \$quant, \$annot, \$counts, \$sampleSetup, \$quantNotes, \$notes, or (if separateAnnot=FALSE) data.frame with annotation- and main quantification-content. If sdrf information has been found, an additional list-element setup will be added containing the entire meta-data as setup\$meta and the suggested organization as setup\$lev.

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, \$notes and optional setup for meta-data from sdrf; or a data.frame with quantitation and annotation if separateAnnot=FALSE

See Also

[read.table](#), [normalizeThis](#) , [readProteomeDiscovererFile](#); [readProlineFile](#) (and other import-functions), [matrixNAinspect](#)

Examples

```
path1 <- system.file("extdata", package="wrProteo")
# Here we'll load a short/trimmed example file (originating from Compomics)
fiNa <- "tinyWombCompo1.csv.gz"
dataWB <- readWombatNormFile(file=fiNa, path=path1, tit="tiny Wombat/Compomics, Normalized ")
summary(dataWB$quant)
```

removeSampleInList	<i>Remove Samples/Columns From List Of Matrixes</i>
--------------------	---

Description

Remove samples (ie columns) from every instance of list of matrixes, like from objects created with proteomics import-functions from this package. Note: This function assumes same order of columns in list-elements 'listElem' !

Usage

```
removeSampleInList(
  dat,
  remSamp,
  listElem = c("raw", "quant", "counts", "sampleSetup"),
  silent = FALSE,
  debug = 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
debug	(logical) display additional messages for debugging
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(raw=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 x\$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,
  debug = 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
debug	(logical) display additional messages for debugging
callFrom	(character) allows easier tracking of messages produced

Value

This function returns a list (like as input), but with missing elements of \$annot filled by information from the columns specified in annCol using files 2 (and 3)

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)
```

shortSoftwName

Get Short Names Of Proteomics Quantitation Software

Description

Get/convert short names of various proteomics quantitation software names for software results handed by this package. A 2-letter abbreviation will be returned

Usage

```
shortSoftwName(
  x,
  tryAsLower = TRUE,
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
```

Arguments

x	(character) software (full) name
tryAsLower	(logical) include lower-caps writing to search
silent	(logical) suppress messages
debug	(logical) additional messages for debugging
callFrom	(character) allows easier tracking of messages produced

Details

So far this function recognizes the following software names: "DIA-NN", "ProteomeDiscoverer", "Compomics", "MaxQuant", "Proline", "TPP", "FragPipe", "MassChroQ", "OpenMS", "Ionbot" and "Sage"

Value

This function returns a vector with 2-letter abbreviation for the software

See Also

[readMaxQuantFile](#)

Examples

```
shortSoftwName(c("maxquant", "DIANN"))
```

summarizeForROC

Summarize Statistical Test Result For Plotting ROC-Curves

Description

This function takes statistical testing results (obtained using [testRobustToNAimputation](#) or [moderTest2grp](#), based on [limma](#)) and calculates specificity and sensitivity values for plotting ROC-curves along a panel of thresholds. Based on annotation (from test\$annot) with the user-defined column for species (argument 'spec') the counts of TP (true positives), FP (false positives), FN (false negatives) and TN are determined. In addition, an optional plot may be produced.

Usage

```

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

```

Arguments

test	(list or 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 statistical test-result to be used for sensitivity and specificity calculations (eg 'BH','lfr' 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 specificity and sensitivity
columnTest	deprecated, please use 'useComp' instead
FCthrs	(numeric) Fold-Change threshold (display as line) give as Fold-change and NOT as log2(FC), default at 1.5, set to NA for omitting
spec	(character) labels for those species which should be matched to column annotCol ('spec') of test\$annot and used for sensitivity and specificity calculations. Important : 1st entry for species designed as constant (ie matrix) and subsequent labels for spike-ins (expected variable)
annotCol	(character, length=1) 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

batchMode	(logical) if batchMode=TRUE the function will return an empty matrix if no proteins qualify for computing ROC (eg all spike-proteins not passing filters), and plotROC will be set to FALSE
tit	(character) optional custom title in graph
color	(character or integer) color in graph
plotROC	(logical) toggle plot on or off
pch	(integer) type of symbol to be used (see par)
bg	(character) background in plot (see par)
overlPlot	(logical) overlay to existing plot if TRUE
silent	(logical) suppress messages
debug	(logical) additional messages for debugging
callFrom	(character) allows easier tracking of messages produced

Details

Determining TP and FP counts requires 'ground truth' experiments, where it is known in advance which proteins are expected to change abundance between two groups of samples. Typically this is done by mixing proteins of different species origin, the first species noted by argument 'spec' designates the species to be considered constant (expected as FN in statistical tests). Then, one or multiple additional spike-in species can be defined. As the spike-in concentration should have been altered between different groups of samples, they are expected as TP.

The main aim of this function consists in providing specificity and sensitivity values, plus counts of TP (true positives), FP (false positives), FN (false negatives) and TN (true negatives), along various thresholds (specified in column 'alpha') for statistical tests performed prior to calling this function.

Note, that the choice of species-annotation plays a crucial role when the counting results are obtained. In case of multiple spike-in species the user should pay attention if they all are expected to change abundance at the same ratio. If not, it is advised to run this function multiple times separately only with the subset of those species expected to change at same ratio.

The dot on the plotted curve shows the results at the level of the single threshold $\alpha=0.05$. For plotting multiple ROC curves as overlay and additional graphical parameters/options you may use [plotROC](#).

See also [ROC on Wikipedia](#) for explanations of TP,FP,FN and TN as well as examples. Note that numerous other packages also provide support for building and plotting ROC-curves : Eg [rocPkgShort](#), [ROCR](#), [pROC](#) or [ROCit](#)

Value

This function returns a numeric matrix containing the columns 'alpha', 'spec', 'sens', 'prec', 'accr', 'FD' plus two columns with absolute numbers of lines (genes/proteins) passing the current threshold level alpha (1st species, all other species)

See Also

replot the figure using [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(Species=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"), annotCol="Species"))
```

test2grp

t-test Each Line Of 2 Groups Of Data

Description

This function performs t-test on two groups of data using **limma**, this is a custom implementation of **moderTest2grp** for proteomics. The final object 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 computations considerably).

Usage

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

Arguments

dat	(matrix or data.frame) main data (may contain NAs)
questNo	(integer) specify here which question, ie comparison should be addressed
useCol	(integer or character)
grp	(character or factor)
annot	(matrix or data.frame)
ROTSn	(integer) number of iterations ROTS runs (stabilization of results may be seen with >300)
silent	(logical) suppress messages
debug	(logical) display additional messages for debugging
callFrom	(character) allow easier tracking of message(s) 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

Pairwise Testing Robust To NA-Imputation

Description

This function 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 recorded 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.

Usage

```
testRobustToNAimputation(
  dat,
  gr = NULL,
  useComparison = NULL,
  annot = NULL,
  retnNA = TRUE,
  avSd = c(0.15, 0.5),
  avSdH = NULL,
  plotHist = FALSE,
  xLab = NULL,
```



```

    tit = NULL,
    imputMethod = NULL,
    seedNo = NULL,
    multCorMeth = c("lfdr", "FDR"),
    nLoop = 100,
    lfdrInclude = NULL,
    ROTSn = NULL,
    pwSep = NULL,
    param = NULL,
    silent = FALSE,
    debug = FALSE,
    callFrom = NULL
  )

```

Arguments

<code>dat</code>	(matrix or data.frame) main data (may contain NA); if <code>dat</code> is list containing <code>\$quant</code> and <code>\$annot</code> as matrix, the element <code>\$quant</code> will be used
<code>gr</code>	(character or factor) replicate association; if <code>dat</code> contains a list-element <code>\$sampleSetup\$groups</code> or <code>\$sampleSetup\$lev</code> this may be used in case <code>gr=NULL</code>
<code>useComparison</code>	(list, character or matrix) optional argument allowing to specify which pairwise comparisons should be performed, default <code>useComparison=NULL</code> will run all pairwise comparisons; may be character vector of pairwise (combined) group-names (matching groups from argument <code>grp</code>), the separator gets automatically determined (if not provided as <code>useComparison\$sep</code>) (eg givingh 'A-B') or matrix where the rownames design the elements to be compared as pairwise; It is also possible to give a list with <code>\$sep</code> (separator to be used when combining) and <code>\$useComparison</code> (regular comparisons) Note : the names of the groups may not contain any '-' to avoid confucing them with pairwise separators !
<code>annot</code>	(matrix or data.frame) annotation (lines must match lines of data !), if <code>annot</code> is <code>NULL</code> and argument <code>dat</code> is a list containing both <code>\$quant</code> and <code>\$annot</code> , the element <code>\$annot</code> will be used
<code>retnNA</code>	(logical) retain and report number of NA
<code>avSd</code>	(numerical,length=2) population characteristics (mean and sd) for >1 NA-neighbours (per line)
<code>avSdH</code>	deprecated, please use <code>avSd</code> instead; (numerical,length=2) population characteristics 'high' (mean and sd) for >1 NA-neighbours (per line)
<code>plotHist</code>	(logical) additional histogram of original, imputed and resultant distribution (made using matrixNANeighbourImpute)
<code>xLab</code>	(character) custom x-axis label
<code>tit</code>	(character) custom title
<code>imputMethod</code>	(character) choose the imputation method (may be 'mode2'(default), 'mode1', 'datQuant', 'modeAdopt', 'informed' or 'none', for details see matrixNANeighbourImpute)
<code>seedNo</code>	(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) deprecated, please used multCorMeth instead (include lfdr estimations, may cause warning message(s) concerning convergence if few too lines/proteins in dataset tested).
ROTSn	(integer) deprecated, please used multCorMeth instead (number of repeats by ROTS, if NULL ROTS will not be called)
pwSep	(character) custom separator to be (checked +) used for pairwise column-labels
param	(list) alternativbe way for passing multiple arguments the same time, will override individual arguments; may contain \$filter (with \$abundThr, \$maxGrpMiss, \$ratMaxNA, \$minSpeNo, \$minTotNo) and \$NAimpute (with \$avSd, \$imputeMethod), \$stat (with \$nLoop, \$lfdr, \$pwSep)
silent	(logical) suppress messages
debug	(logical) additional messages for debugging
callFrom	(character) This function allows easier tracking of messages produced

Details

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 summarized afterwards by median over all repeated runs to remove 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 optionally be included, if desired. This function returns a **limma**-like S3 list-object further enriched by additional fields/elements.

The argument *multCorMeth* allows to choose which multiple correction algorithms 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' refers to Benjamini-Yakuteli FDR, 'tValTab' allows exporting all individual t-values from the repeated NA-substitution and subsequent testing.

This function is compatible with automatic extraction of experimental setup based on *sdrf* or other quantitation-specific sample annotation. In this case, the results of automated importing and mining of sample annotation should be stored as *\$sampleSetup\$groups* or *\$sampleSetup\$lev*

It is possible to limit the pairwise combinations to a custom designed set using the argument *useComparison*. This may be a matrix where very line designs a new pairwise comparison and the first column is refers to 'sample' while the second column assigns the 'reference'. Otherwise, one may provide a character vector as *useComparison* where each entry refers to a new pairwise comparison. In this case it is recommended to use '-' as separator for combining the group-names to be used as 'sample' and 'reference'. In case the names of groups (argument *grp*) do not contain any '-' the single character separator '-' may also be used in compatibility with use in bioconductor packae **limma**. However, if *grp*) does contain any '-', results will be presented using '-' as separator to prevent any confusion.

Value

This function returns a limma-type MA-object ('MArrayLM', can be handled like a list). For details 'on choice of NA-imputation procedures with arguments 'imputeMethod' and 'avSd' please see [matrixNANeighbourImpute](#).

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

See Also

NA-imputation via [matrixNANeighbourImpute](#), moderated t-test without NA-imputation [moderTest2grp](#), calculating lfr [pVal2lfr](#), 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      # raise lines 13:16
dat1[19:20,1:3] <- dat1[19:20,1:3] +3      # raise lines 19:20
dat1[15:18,4:6] <- dat1[15:18,4:6] +1.4    # raise lines 15:18
dat1[dat1 <1] <- NA                        # mimick some NAs for low abundance
## normalize data
boxplot(dat1, main="Data Before Normalization", las=1)
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 imputations (may take >10 sec, >60 sec if ROTSn >0 !)
PLtestR1 <- testRobustToNAimputation(dat=dat1, gr=grp1, retnNA=TRUE, nLoop=20)
names(PLtestR1)
head(PLtestR1$p.value)
head(PLtestR1$BH)
head(PLtestR1$means)
boxplot(PLtestR1$datImp, main="Data At Normalization & Imputation", las=1)

## custom selection of comparisons (incl custom orientation)
useComp <- c("A-B", "B-A")      # You can choose orientation sample/reference
PLtestR2 <- testRobustToNAimputation(dat=dat1, gr=grp1, useComparison=useComp,
  retnNA=TRUE, nLoop=20)
head(PLtestR2$BH)
head(PLtestR2$means)
```

Description

Please use VolcanoPlotW() from package wrGraph. This function does NOT produce a plot any more.

Usage

```
VolcanoPlotW2(
  Mvalue,
  pValue = NULL,
  useComp = 1,
  filtFin = NULL,
  ProjNa = NULL,
  FCthrs = NULL,
  FdrList = NULL,
  FdrThrs = NULL,
  FdrType = NULL,
  subTxt = NULL,
  grayIncr = TRUE,
  col = NULL,
  pch = 16,
  compNa = NULL,
  batchFig = FALSE,
  cexMa = 1.8,
  cexLa = 1.1,
  limM = NULL,
  limp = NULL,
  annotColumn = NULL,
  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 columns 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())

<code>filtFin</code>	(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 <code>filtFin</code> element will be applied
<code>ProjNa</code>	(character) custom title
<code>FCthrs</code>	(numeric) Fold-Change threshold (display as line) give as Fold-change and NOT $\log_2(FC)$, default at 1.5, set to NA for omitting
<code>FdrList</code>	(numeric) FDR data or name of list-element
<code>FdrThrs</code>	(numeric) FDR threshold (display as line), default at 0.05, set to NA for omitting
<code>FdrType</code>	(character) FDR-type to extract if Mvalue is 'MArrayLM'-object (eg produced by from <code>moderTest2grp</code> etc); if NULL it will search for suitable fields/values in this order : 'FDR', 'BH', "lfdr" and 'BY'
<code>subTxt</code>	(character) custom sub-title
<code>grayIncr</code>	(logical) if TRUE, display overlay of points as increased shades of gray
<code>col</code>	(character) custom color(s) for points of plot (see also par)
<code>pch</code>	(integer) type of symbol(s) to plot (default=16) (see also par)
<code>compNa</code>	(character) names of groups compared
<code>batchFig</code>	(logical) if TRUE figure title and axes legends will be kept shorter for display on fewer space
<code>cexMa</code>	(numeric) font-size of title, as expansion factor (see also <code>cex</code> in par)
<code>cexLa</code>	(numeric) size of axis-labels, as expansion factor (see also <code>cex</code> in par)
<code>limM</code>	(numeric, length=2) range of axis M-values
<code>limp</code>	(numeric, length=2) range of axis FDR / p-values
<code>annotColumn</code>	(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 preferred text for annotating the best points (if <code>namesNBest</code> allows to do so.)
<code>annColor</code>	(character or integer) colors for specific groups of annotation (only if Mvalue is MArrayLM-object containing matrix \$annot)
<code>cexPt</code>	(numeric) size of points, as expansion factor (see also <code>cex</code> in par)
<code>cexSub</code>	(numeric) size of subtitle, as expansion factor (see also <code>cex</code> in par)
<code>cexTxLab</code>	(numeric) size of text-labels for points, as expansion factor (see also <code>cex</code> in par)
<code>namesNBest</code>	(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 <code>annotColumn</code> will be used as text
<code>NbestCol</code>	(character or integer) colors for text-labels of best points
<code>sortLeg</code>	(character) sorting of 'SpecType' annotation either ascending ('ascend') or descending ('descend'), no sorting if NULL
<code>NaSpecTypeAsContam</code>	(logical) consider lines/proteins with NA in Mvalue\$annot["SpecType"] as contaminants (if a 'SpecType' for contaminants already exists)

useMar	(numeric,length=4) custom margins (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

Value

deprecated - returns nothing

See Also

this function was replaced by [plotPCAw](#))

Examples

```
set.seed(2005); mat <- matrix(round(runif(900),2), ncol=9)
```

writeFasta2	<i>Write Sequences In Fasta Format To File This function writes sequences from character vector or parsed matrix to file as fasta formatted file (close to Rhrefhttps://www.uniprot.orgUniProt) 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.</i>
-------------	--

Description

Write Sequences In Fasta Format To File

This function writes sequences from character vector or parsed matrix to file as fasta formatted file (close to [UniProt](#)) 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,
  sep = "|",
  suplSep = " ",
  suplColumns = c("ProteinName", "OS=", "OX=", "GN=", "PE=", "SV="),
  ref = NULL,
  lineLength = 60,
  eol = "\n",
  truSuf = "_tru",
  substName = "protein",
```

```

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

```

Arguments

<code>prot</code>	(character) vector of sequences, names will be used for fasta-header
<code>fileNa</code>	(character) name (and path) for file to be written
<code>sep</code>	(character) primary separator for fasta-header (when <code>prot</code> is matrix)
<code>suplSep</code>	(character) supplemental separator for additional UniProt fields
<code>suplColumns</code>	(character) column names to be used for supplemental separator (for additional UniProt fields)
<code>ref</code>	(character) optional/additional set of (reference-) sequences (only for comparison to <code>prot</code>), length of proteins from <code>prot</code> will be checked to mark truncated proteins by <code>'_tru'</code>
<code>lineLength</code>	(integer, length=1) number of sequence characters per line (default 60, should be >1 and <10000)
<code>eol</code>	(character) the character(s) to print at the end of each line (row); for example, <code>eol = "\r\n"</code> will produce Windows' line endings on a Unix-alike OS
<code>truSuf</code>	(character) suffix to be added for sequences found truncated when comparing with <code>ref</code>
<code>substName</code>	(character, length=1) in case no names/elements are given to be used as fasta-headers this term will be enumerated
<code>silent</code>	(logical) suppress messages
<code>debug</code>	(logical) supplemental messages for debugging
<code>callFrom</code>	(character) allows easier tracking of messages produced

Details

If a named character vector is entered as `prot`, names will be used as annotation of the sequence entries. Sequences without any names will be given generic headers like `protein01 ...` etc.

If `prot` is a matrix the column named `'sequence'` or `'seq'` will be considered as sequence and the rest will get concatenated to form the fasta-header using the separator given by argument `sep`

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

```

prot <- c(SEQU1="ABCDEFGHijkl", SEQU2="CDEFGHIJKLMNOP")
writeFasta2(prot, fileNa=file.path(tempdir(),"testWrite.fasta"), lineLength=6)

```

Index

.atomicMasses, [4](#)
.checkKnitrProt, [4](#)
.checkSetupGroups, [5](#)
.commonSpecies, [6](#), [8](#)
.extrSpecPref, [7](#)
.getSpeciesFromEntryName, [8](#)
.imputeNA, [9](#)
.parseFastaHeader, [10](#)
.plotQuantDistr, [11](#)

AAMass, [12](#), [18](#)
adjustUnitPrefix, [81](#)
AucROC, [13](#), [94](#)

checkUnitPrefix, [81](#)
cleanListCoNames, [14](#)
combineMultFilterNAimput, [15](#)
combineRedundLinesInList, [30](#)
convAASeq2mass, [18](#)
convToNum, [13](#), [18](#), [34](#)
corColumnOrder, [19](#)
countNoOfCommonPeptides, [20](#)

exportAsWombatP, [21](#)
exportSdrfDraft, [23](#)
extractTestingResults, [24](#)
extrSpeciesAnnot, [27](#)

foldChangeArrow, [28](#), [29](#)
foldChangeArrow2, [28](#)
fuseProteomicsProjects, [29](#)

getUPSlacc, [31](#), [32](#)
grep, [15](#), [27](#)

hist, [36](#)

inspectSpeciesIndic, [32](#)
isolNAneighb, [33](#)

MAplotW, [29](#)

massDeFormula, [4](#), [13](#), [18](#), [34](#)
matrixNAinspect, [35](#), [45](#), [62](#), [89](#)
matrixNAneighbourImpute, [9](#), [33](#), [36](#), [97](#), [99](#)
moderTest2grp, [20](#), [22](#), [24](#), [26](#), [40](#), [92](#), [94–96](#),
[99](#)
moderTestXgrp, [20](#), [22](#), [24](#), [26](#)

na.fail, [33](#), [36](#), [38](#)
naOmit, [36](#)
normalizeThis, [43](#), [45–48](#), [50](#), [52](#), [54](#), [57](#), [59](#),
[60](#), [62](#), [64–67](#), [69](#), [72–74](#), [76](#), [77](#), [79](#),
[87](#), [89](#)

par, [12](#), [40](#), [94](#), [101](#), [102](#)
plotPCAw, [102](#)
plotROC, [14](#), [39](#), [94](#)
presenceFilt, [5](#), [16](#), [17](#), [42](#)
pVal2lfdr, [96](#), [99](#)

razorNoFilter, [41](#)
read.table, [45](#), [47](#), [50](#), [54](#), [59](#), [62](#), [65](#), [67](#), [70](#),
[73](#), [76](#), [79](#), [89](#)
readAlphaPeptFile, [42](#)
readDiaNNFile, [8](#), [45](#)
readDiaNNPeptides, [48](#)
readFasta2, [10](#), [21](#), [50](#), [103](#)
readFragpipeFile, [6](#), [8](#), [12](#), [52](#), [73](#), [76](#)
readIonbotPeptides, [54](#)
readLines, [10](#), [51](#)
readMassChroQFile, [57](#)
readMaxQuantFile, [6](#), [8](#), [12](#), [20](#), [22–24](#), [47](#),
[50](#), [54](#), [59](#), [65](#), [67](#), [73](#), [76](#), [79](#), [81](#), [91](#),
[92](#)
readMaxQuantPeptides, [63](#)
readOpenMSFile, [66](#)
readProlineFile, [6](#), [8](#), [12](#), [45](#), [47](#), [50](#), [54](#), [59](#),
[62](#), [65](#), [67](#), [68](#), [73](#), [76](#), [89](#), [91](#)
readProtDiscovererPeptides, [71](#)
readProtDiscovFile, [47](#), [50](#), [54](#), [67](#), [71](#), [91](#)

readProteomeDiscovererFile, [20](#), [22](#), [45](#),
 [62](#), [71](#), [74](#), [79](#), [89](#)
readProteomeDiscovererPeptides, [71](#), [77](#)
readSampleMetaData, [79](#), [82](#), [83](#)
readSdrf, [81](#), [82](#)
readUCSCtable, [83](#), [85](#)
readUniProtExport, [84](#), [84](#)
readWombatNormFile, [86](#)
removeSampleInList, [89](#)
replicateStructure, [81–83](#)
replMissingProtNames, [90](#)

sd, [30](#)
shortSoftwName, [91](#)
stableMode, [33](#), [38](#)
summarizeForROC, [13](#), [14](#), [39](#), [40](#), [92](#)

t.test, [94](#), [96](#), [99](#)
test2grp, [93](#), [94](#), [95](#)
testRobustToNAimputation, [26](#), [33](#), [38](#), [90](#),
 [92–94](#), [96](#)

vioplotW, [47](#), [49](#), [53](#), [58](#), [67](#), [69](#), [73](#), [76](#), [78](#)
VolcanoPlotW, [29](#)
VolcanoPlotW2, [99](#)

writeFasta2, [10](#), [51](#), [102](#)