Package ‘wrProteo’

June 16, 2023

Version 1.10.0
Title Proteomics Data Analysis Functions
Author Wolfgang Raffelsberger [aut, cre]
Maintainer Wolfgang Raffelsberger <w.raffelsberger@gmail.com>
Description Data analysis of proteomics experiments by mass spectrometry is supported by this collection of functions mostly dedicated to the analysis of (bottom-up) quantitative (XIC) data. Fasta-formatted proteomes (eg from UniProt Consortium <doi:10.1093/nar/gky1049>) can be read with automatic parsing and multiple annotation types (like species origin, abbreviated gene names, etc) extracted.

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>), DiaNN (Demichev et al 2020 <doi:10.1038/s41592-019-0638-x>),
- OpenMS (<doi:10.1038/nmeth.3959>), ProteomeDiscoverer, Proline (Bouyssie et al 2020 <doi:10.1093/bioinformatics/btaa118>) and Wombat-P.

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>) 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> can be run, enhanced by multiple rounds of NA-replacements to provide robustness towards rare stochastic events.

Multi-species samples, as frequently used in benchmark-tests (eg Navarro et al 2016 <doi:10.1038/nbt.3685>, Ramus et al 2016 <doi:10.1016/j.jprot.2015.11.011>), can be run with special options considering
such sub-groups during normalization and testing. Subsequently, ROC curves (Hand and Till 2001 <doi:10.1023/A:1010920819831>) can be constructed to compare multiple analysis approaches.

As detailed example the data-set from Ramus et al 2016 <doi: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 (>= 1.12.0)
**Suggests**  data.table, fdrtool, kableExtra, MASS, RColorBrewer, readxl, ROTS, rmarkdown, R.utils, sm, wrGraph (>= 1.3.0)
**License**  GPL-3
**Encoding**  UTF-8
**VignetteBuilder**  knitr
**RoxygenNote**  7.2.3
**NeedsCompilation**  no
**Repository**  CRAN
**Date/Publication**  2023-06-16 08:10:03 UTC

**R topics documented:**

.atomicMasses ......................................................... 3
.checkKnitrProt ...................................................... 4
.checkSetupGroups ..................................................... 4
.commonSpecies ....................................................... 5
.extrSpecPref ......................................................... 6
.imputeNA ............................................................. 7
.plotQuantDistr ....................................................... 8
.AAmass ............................................................... 9
.AucROC ............................................................... 10
.cleanListCoNames .................................................... 11
.combineMultFilterNAimput ......................................... 12
.convAASeq2mass ...................................................... 14
.corColumnOrder ...................................................... 15
.countNoOfCommonPeptides .......................................... 16
.exportSdrfDraft ..................................................... 17
.extractTestingResults ............................................... 19
.extrSpeciesAnnot .................................................... 20
.foldChangeArrow2 .................................................... 21
.isolNAneighb ....................................................... 23
.massDeFormula ....................................................... 24
.matrixNAinspect ...................................................... 25
.matrixNAinspect ...................................................... 25
.plotROC ............................................................. 29
.razorNoFilter ....................................................... 30
.atomicMasses

Molecular mass for Elements

Description

This function returns the molecular mass based on main elements found in biology/proteomics as average and mono-isotopic mass. The result includes H, C, N, O, P, S, Se and the electron. The values are based 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()

---

.checkSetupGroups  Additional/final chek and adjustments to sample-order after readSampleMetaData()

---

Description

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

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

Arguments

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

Value

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

See Also

used in readProtDiscovererFile, readMaxQuantFile, readProlineFile, readFragpipeFile

Examples

set.seed(2021)

---

.commonSpecies

Get matrix with UniProt abbreviations for common species

Description

This (low-level) function allows accessing matrix with UniProt abbreviations for common species
This information maybe used to harmonize species descriptions.
Usage

.commonSpecies()

Value

This function returns a 2-column matrix with species names

See Also

used e.g. in readProtDiscovererFile, readMaxQuantFile, readProlineFile, readFragpipeFile

Examples

.commonSpecies()

---

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

Arguments

specPref (list) may contain $mainSpecies, $conta ...
annot (matrix) main protein annotation
useColumn (factor) columns from annot to use/mine
suplInp (matrix) additional custom annotation
silent (logical) suppress messages
debug (logical) display additional messages for debugging (starting with 'mainSpecies', 'conta'
and others - later may overwrite prev settings)
callFrom (character) allow easier tracking of messages produced
This function returns a matrix with additional column 'SpecType'

See Also

used in readProtDiscovererFile, readMaxQuantFile, readProlineFile, readFragpipeFile

Examples

.checkKnitrProt()

Description

This (lower-level) function allows to perform the basic NA-imputation. 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

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

Value

This function returns a list with $data and $datImp
.plotQuantDistr

Generic plotting of density distribution for quantitation import-functions

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

Arguments

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

See Also

for more complex treatment matrixNAneighbourImpute:

Examples

dat1 <- matrix(11:22, ncol=4)
.imputeNA(dat1, impParam=c(mean(dat1, na.rm=TRUE), 0.1))
AAmass

refLi (integer) to display number reference lines
refLiIni (integer) to display initial number reference lines
tit (character) custom title
las (integer) indicate orientation of text in axes
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("li",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")

---

AAmass

Molecular mass for amino-acids

Description

Calculate molecular mass based on atomic composition

Usage

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

Arguments

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

Value

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

AUC from ROC-curves

Description

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

Usage

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

Arguments

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

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 perfomed before character substitutions by subst)
subst (character of length=2, or matrix with 2 columns) pair(s) of character-strings for replacement (1st as search-item and 2nd as replacement); this part is performed after character-removal via rem
lstE (character, length=1) names of list-elements where colnames should be cleaned
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(raw=dat1, quant=dat1, notes="other..")
cleanListCoNames(dat1, rem=c("sample_"), c("-","-"))

---

combineMultFilterNAimput

Combine Multiple Filters On NA-imputed Data

Description

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

Usage

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

```
callFrom = NULL
```

**Arguments**

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

**Value**

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

**See Also**

- `presenceFilt`

**Examples**

```r
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)
```
Molecular mass for amino-acids

Description

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

Usage

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

Arguments

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

Value

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

See Also

- `massDeFormula`, `AAmass`, `convToNum`

Examples

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

Order Columns in list of matrixes

Description

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

Usage

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

Arguments

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

Value

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

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

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

See Also

moderTestXgrp for single comparisons; order

Examples

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

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

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
debag (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 matrices for each peptide (as line) and unique/shared/etc for each species; $byProt as list of matrices with count data per protein (as line) for each species; $tab with simple summary-type count data

See Also

readFasta2 and/or cleave-methods in package cleaver

Examples

```r
## The example mimics a proteomics experiment where extracts from E coli and Saccharomyces cerevisiae were mixed, thus not all peptides may occur unique.
(mi2 = countNoOfCommonPeptides(Ec=list(E1=letters[1:4],E2=letters[3:7]],
    E3=letters[4,8,13]],E4=letters[9]],Sc=list(S1=letters[2:3,6]],S2=letters[10:13]],S3=letters[5,6,11]])
## 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
```

Description

Sample/experimental annotation meta-data form MaxQuant that was previously import can now be formatted in sdrf-style and exported using this function to write a draft-sdrf-file. Sdrf-files provide additional meta-information about samples 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

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

Details

Gathering information about samples and MS-runs requires that at data-import the additioal files created from software, like MaxQuant, was present and imported. 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 will be further developed to gather equivalent information in the future.

Value

This function writes an Sdrf draft to file

See Also

This function may be used after reading/importig 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, sup1AnnotFile=TRUE)
## Here we'll write simply in the current temporary directory of this R-session
extractTestingResults

```r
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 `moderTest2grp` or `moderTestXgrp`. The user can define the threshold which type of multiple testing correction should be used (as long as the multiple testing correction method was actually performed as part of testing).

#### Usage

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

#### Arguments

- **stat** (`MArrayLM`-object or list) Designed for the output from `moderTest2grp` or `moderTestXgrp`
- **compNo** (integer) the comparison number/index to be used
- **statTy** (character) the multiple-testing correction type to be considered when looking for significant changes with threshold `thrsh` (depends on which have been run initially with `moderTest2grp` or `moderTestXgrp`)
- **thrsh** (numeric) the threshold to be applied on `statTy` for the result of the 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 when returning results
- **addTy** (character) additional groups to add (so far only "allMeans" available) in addition to the means used in the pairwise comparison
extrSpeciesAnnot

Extract species annotation

Description

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

Usage

eextrSpeciesAnnot(
  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
grep

Examples

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

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

Description

NOTE : This function is deprecated, please use foldChangeArrow instead !! This function was made for adding an arrow indicating a fold-change to MA- or Volcano-plots. When comparing multiple concentrations of standards in benchmark-tests it may be useful to indicate the expected ratio in a pair-wise comparison. In case of main input as list or MArrayLM-object (as generated from limma), the column-names of multiple pairwise comparisons can be used for extracting a numeric content (supposed as concentrations in sample-names) which will be used to determine the expected ratio used for plotting. Optionally the ratio used for plotting can be returned as numeric value.
foldChangeArrow2

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 elements side and adj just will be redefined to accommodate the text in the corner specified.

Ultimately this function will be integrated to the package wrGraph.
isolNAneighb

Value

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

See Also

new version: foldChangeArrow; used with MAplotW, VolcanoPlotW

Examples

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

isolNAneighb(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 matrixNAneighbourImpulse and testRobustToNAimputation; estimation of mode stableMode; detection of NAs na.fail
massDeFormula

Molecular mass from chemical formula

Description
Calculate molecular mass based on atomic composition

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

Arguments

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

Value
This function returns a numeric vector with mass

See Also
convToNum

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

matrixNAinspect makes histograms of the full data and shows sub-population of NA-neighbour values. The aim of this function is to investigate the nature of NA values in matrix (of experimental measures) where replicate measurements are available. If a given element was measured twice, and one of these measurements revealed a NA while the other one gave a (finite) numeric value, the non-NA-value is considered a NA-neighbour. The subpopulation of these NA-neighbour values will then be highlighted in the resulting histogram. In a number of experimental 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 measurements 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.

Usage

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

Arguments

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

Value

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

See Also

hist, na.fail, naOmit

Examples

```r
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))
```

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

Usage

```r
matrixNAneighbourImpute(
    dat,
    gr,
    imputMethod = "mode2",
    retnNA = TRUE,
    avSd = c(0.15, 0.5),
```
matrixNAneighbourImpute

avSdH = NULL,
NAneigLst = NULL,
plotHist = c("hist", "mode"),
xLab = NULL,
xLim = NULL,
yLab = NULL,
yLim = NULL,
tit = NULL,
figImputDetail = TRUE,
seedNo = NULL,
silent = FALSE,
callFrom = NULL,
debug = FALSE
)

Arguments

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

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

There are a number of experimental settings where low measurements may be reported as NA. Sometimes an arbitrary defined baseline (as 'zero') may provoke those values found below being unfortunately reported as NA or as 0 (in case of MaxQuant). In quantitative proteomics (DDA-mode) the presence of numerous high-abundance peptides will lead to the fact that a number of less intense MS-peaks don’t get identified properly and will then be reported as NA in the respective samples, while the same peptides may by correctly identified and quantified in other (replicate) samples. So, if a given protein/peptide gets properly quantified in some replicate samples but reported as NA in other replicate samples one may thus speculate that similar values like in the successful quantifications may have 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 plots ROC curves based on results from `summarizeForROC`. This function plots only, it does not return any data. It allows printing simultaneously multiple ROC curves from different studies, it is also compatible with data from 3 species mix as in proteomics benchmark. Input can be prepared using `moderTest2grp` followed by `summarizeForROC`.

**Usage**

```r
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**

- `dat` (matrix) from testing (eg `summarizeForROC`)
- `...` optional additional data-sets to include as separate ROC-curves to same plot (must be of same type of format as `dat`)
- `useColumn` (integer or character, length=2) columns from `dat` to be used for specificity and sensitivity
- `methNames` (character) names of methods (data-sets) to be displayed
- `col` (character) custom colors for lines and text (choose one color for each different data-set)
razorNoFilter

pch (integer) type of symbol to be used (see also par)
bg (character) background color in plot (see also par)
tit (character) custom title
xlim (numeric, length=2) custom x-axis limits
ylim (numeric, length=2) custom y-axis limits
point05 (numeric) specific point to highlight in plot (typically at alpha=0.05)
pointSi (numeric) size of points (as expansion factor cex)
nByMeth (integer) value of n to display
speciesOrder (integer) custom order of species in legend
txtLoc (numeric, length=3) location for text (x, y and proportional factor for line-offset, default is c(0.4,0.3,0.04))
legCex (numeric) cex expansion factor for legend (see also par)
las (numeric) factor for text-orientation (see also par)
addSuplT (logical) add text with information about precision, accuracy and FDR
silent (logical) suppress messages
debug (logical) display additional messages for debugging
callFrom (character) allow easier tracking of message(s) produced

Value
This function returns only a plot with ROC curves

See Also
summarizeForROC, moderTest2grp

Examples
roc0 <- cbind(alph=c(2e-6,4e-5,4e-4,2.7e-3,1.6e-2,4.2e-2,8.3e-2,1.7e-1,2.7e-1,4.1e-1,5.3e-1, 6.8e-1,1.9e-1,9.7e-1), spec=c(1,1,1,0.957,0.915,0.809,0.702,0.489,0.362,0.234, 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(0,0,0,2,4,9,14,24,36,41) )
plotROC(roc0)

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

Description
razorNoFilter filters based on either a) number of total peptides and specific peptides or b) number of razor peptides. 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).
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 specific peptides

minTotNo (integer) minimum total i.e. max razor number of peptides

silent (logical) suppress messages

debug (logical) display additional messages for debugging

callFrom (character) allow easier tracking of messages produced

Value

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

See Also

`presenceFilt`

Examples

```r
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")
```
readDiaNNFile

**Description**

This function allows importing protein identification and quantification results from DIA-NN, see also doi:10.1038/s415920190638xDemichev et al, 2020. Data should be exported as tabulated text (tsv) as protein-groups (pg) to allow import by thus 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 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**

readDiaNNFile(
  fileName,
  path = NULL,
  normalizeMeth = "median",
  sampleNames = NULL,
  read0asNA = TRUE,
  quantCol = "\.aw$",
  annotCol = NULL,
  refLi = NULL,
  separateAnnot = TRUE,
  FDRCol = NULL,
  groupPref = list(lowNumberOfGroups = 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
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>read0asNA</td>
<td>(logical) decide if initial quantifications at 0 should be transformed to NA (thus avoid -Inf in log2 results)</td>
</tr>
<tr>
<td>quantCol</td>
<td>(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</td>
</tr>
<tr>
<td>annotCol</td>
<td>(character) column names to be read/extracted for the annotation section (default c(&quot;Accession&quot;,&quot;Description&quot;,&quot;Gene&quot;,&quot;Contaminant&quot;,&quot;Sum.PEP.Score&quot;,&quot;Coverage...&quot;,&quot;X..Peptides&quot;,&quot;X..AAs&quot;,&quot;MW..kDa.&quot;))</td>
</tr>
<tr>
<td>refLi</td>
<td>(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</td>
</tr>
<tr>
<td>separateAnnot</td>
<td>(logical) if TRUE output will be organized as list with $annot, $abund for initial/raw abundance values and $quant with final log2 (normalized) quantitations</td>
</tr>
<tr>
<td>FDRCol</td>
<td>(list) - not used</td>
</tr>
<tr>
<td>groupPref</td>
<td>(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)</td>
</tr>
<tr>
<td>plotGraph</td>
<td>(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</td>
</tr>
<tr>
<td>titGraph</td>
<td>(character) custom title to plot of distribution of quantitation values</td>
</tr>
<tr>
<td>wex</td>
<td>(integer) relative expansion factor of the violin-plot (will be passed to vioplotW)</td>
</tr>
<tr>
<td>specPref</td>
<td>(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)</td>
</tr>
<tr>
<td>gr</td>
<td>(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from sdrf and/or suplAnnotFile (if provided)</td>
</tr>
<tr>
<td>sdrf</td>
<td>(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</td>
</tr>
<tr>
<td>suplAnnotFile</td>
<td>(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)</td>
</tr>
<tr>
<td>silent</td>
<td>(logical) suppress messages</td>
</tr>
<tr>
<td>debug</td>
<td>(logical) additional messages for debugging</td>
</tr>
<tr>
<td>callFrom</td>
<td>(character) allow easier tracking of messages produced</td>
</tr>
</tbody>
</table>
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 lack in later steps of analysis. It is suggested to rather read protein-group (pg) files.

Using the argument supplAnnotFile 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

diaNNF1 <- "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=diaNNF1, specPref=specPref1, tit="Tiny DIA-NN Data")
summary(dataNN$quant)

readDiaNNPeptides

Read Tabulated Files Exported by DiaNN At Peptide Level

Description

This function allows importing peptide identification and quantification results from DiaNN, see also doi: 10.1038/s41592-019-0638-x Demichev et al, 2020. 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 the elements: $annot, $raw and $quant, or a data.frame with the quantication data and a part of the annotation if argument separateAnnot=FALSE.

Usage

readDiaNNPeptides(
  fileName,
  path = NULL,
  normalizeMeth = "median",
  sampleNames = NULL,
  read0asNA = TRUE,
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..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)
**plotGraph** (logical or integer) optional plot of type violaplot 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
readDiaNNPeptides

wex (integer) relative expansion factor of the violin-plot (will be passed to \texttt{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 \texttt{readSdrf} or a list from \texttt{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
\texttt{read.table, normalizeThis), readMaxQuantFile, readProtDiscovFile, readProlineFile}

Examples

diaNNFil <- "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=diaNNFil1, 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. If `tableOut=TRUE` output may be organized as matrix for separating meta-annotation (e.g., `uniqueIdentifier`, `entryName`, `proteinName`, `GN`) in separate columns.

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

**Arguments**
- `filename` (character) names fasta-file to be read
- `delim` (character) 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 ‘empty’ allows removing entries without any sequence entries; set to ‘duplicated’ to remove duplicate entries (same sequence and same header)
- `tableOut` (logical) toggle to return named character-vector or matrix with enhanced parsing of fasta-header. The resulting matrix will contain the columns ‘database’, ‘uniqueIdentifier’, ‘entryName’, ‘proteinName’, sequence and further columns depending on argument `UniprSep`
- `UniprSep` (character) separators for further separating entry-fields if `tableOut=TRUE`, see also [UniProt-FASTA-headers](http://www.uniprot.org)
- `cleanCols` (logical) remove columns with all entries NA, if `tableOut=TRUE`
- `silent` (logical) suppress messages
- `callFrom` (character) allows easier tracking of messages produced
- `debug` (logical) supplemental messages for debugging
Value

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

See Also

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

Examples

```r
## 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

Read Tabulated Files Exported by FragPipe At Protein Level

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

```r
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), 
  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 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..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)
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)
**readMassChroQFile**

- **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.

- **sup1AnnotFile** (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 `sup1AnnotFile` 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**

```r
FPproFi1 <- "tinyFragpipe1.tsv.gz"
p1th <- system.file("extdata", package="wrProteo")
## let's define the main species and allow tagging some contaminants
specP1f <- c(conta="conta|CON_|LYSC_CHICK", mainSpecies="MOUSE")
dataFP <- readFragpipeFile(p1th, file=FPproFi1, specPref=specPref1, tit="Tiny Fragpipe Data")
summary(dataFP$quant)
```

---

**readMassChroQFile**  
Read tabulated files imported from MassChroQ

**Description**

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

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), 
  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 - made 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)
readMassChroQFile

sdrf (character, list or data.frame) optional extraction and adding of experimental
meta-data: if character, this may be the ID at ProteomeExchange. Besides, the
output from readSdf or a list from defineSamples may be provided; if gr is
provided, it gets priority for grouping of replicates

suplAnnotFile (logical or character) optional reading of supplemental files produced by Pro-
toileDiscoverer; 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 (de-
faults to low number of groups, ie higher number of samples per group)

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 re-
main empty to keep format compatible to other import functions.

This function has been developed using MassChroQ version 2.2 and R-package MassChroqR ver-
sion 0.4.0. Both are distributed by the PAPPSO (http://pappso.inrae.fr/). When saving quantifica-
tions 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)
readMaxQuantFile

Read Quantitation Data-Files (proteinGroups.txt) Produced From MaxQuant At Protein Level

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),
  titGraph = NULL,
  wex = 1.6,
  plotGraph = TRUE,
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)

Arguments

path (character) path of file to be read
### readMaxQuantFile

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fileName</code></td>
<td>(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.</td>
</tr>
<tr>
<td><code>normalizeMeth</code></td>
<td>(character) normalization method, defaults to median, for more details see <code>normalizeThis</code>.</td>
</tr>
<tr>
<td><code>quantCol</code></td>
<td>(character or integer) exact col-names, or if length=1 content of <code>quantCol</code> will be used as pattern to search among column-names for <code>$quant</code> using <code>grep</code>.</td>
</tr>
<tr>
<td><code>contamCol</code></td>
<td>(character or integer, length=1) which columns should be used for contaminants</td>
</tr>
<tr>
<td><code>pepCountCol</code></td>
<td>(character) pattern to search among column-names for count data (1st entry for 'Razor + unique peptides', 2nd fro 'Unique peptides', 3rd for 'MS/MS.count' (PSM))</td>
</tr>
<tr>
<td><code>read0asNA</code></td>
<td>(logical) decide if initial quantifications at 0 should be transformed to NA (thus avoid -Inf in log2 results)</td>
</tr>
<tr>
<td><code>refLi</code></td>
<td>(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</td>
</tr>
<tr>
<td><code>sampleNames</code></td>
<td>(character) custom column-names for quantification data; this argument has priority over <code>suplAnnotFile</code></td>
</tr>
<tr>
<td><code>extrColNames</code></td>
<td>(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')</td>
</tr>
<tr>
<td><code>specPref</code></td>
<td>(character) prefix to identifiers allowing to separate i) recognize contamination database, ii) species of main identifications and iii) spike-in species</td>
</tr>
<tr>
<td><code>remRev</code></td>
<td>(logical) option to remove all protein-identifications based on reverse-peptides</td>
</tr>
<tr>
<td><code>remConta</code></td>
<td>(logical) option to remove all proteins identified as contaminants</td>
</tr>
<tr>
<td><code>separateAnnot</code></td>
<td>(logical) if TRUE output will be organized as list with $annot, $abund for initial/raw abundance values and $quant with final normalized quantitations</td>
</tr>
<tr>
<td><code>gr</code></td>
<td>(character or factor) custom defined pattern of replicate association, will override final grouping of replicates from <code>sdrf</code> and/or <code>suplAnnotFile</code> (if provided)</td>
</tr>
<tr>
<td><code>sdrf</code></td>
<td>(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 <code>readSdrf</code> or a list from <code>defineSamples</code> may be provided; if <code>gr</code> is provided, <code>gr</code> gets priority for grouping of replicates</td>
</tr>
<tr>
<td><code>suplAnnotFile</code></td>
<td>(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 default to files 'summary.txt' (needed to match information of <code>sdrf</code>) and 'parameters.txt' which can be found in the same folder as the main quantitation results; if character the respective file-names (relative ro 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)</td>
</tr>
<tr>
<td><code>groupPref</code></td>
<td>(list) additional parameters for interpreting meta-data to identify structure of groups (replicates), will be passed to <code>readSampleMetaData</code>. 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)</td>
</tr>
</tbody>
</table>
**readMaxQuantFile**

**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`, `$notes`, or (if `separateAnnot=FALSE`) a data.frame with annotation- and main quantification-content. If `sdrf` information has been found, an additional list-element will be added containing the entire meta-data as `$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 imprtf functions), `matrixNAinspect`
Examples

```r
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**  
Read Peptide Identification and Quantitation Data-Files (peptides.txt) Produced By MaxQuant

**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**

```r
readMaxQuantPeptides(
  path,
  fileName = "peptides.txt",
  normalizeMeth = "median",
  quantCol = "LFQ.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,
  supplAnnotFile = NULL,
  groupPref = list(lowNumberOfGroups = TRUE),
  titGraph = NULL,
  wex = 1.6,
  plotGraph = TRUE,
  silent = FALSE,
  debug = FALSE,
)```
readMaxQuantPeptides

callFrom = NULL
)

Arguments

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, i.e. 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 peptide-IDs, default )

specPref (character) prefix to identifiers allowing to separate i) recognition 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 ini- tial/raw abundance values and $quant with final normalized quantitations

ggr (character or factor) custom defined pattern of replicate association, will over- ride final grouping of replicates from sdrf and/or suplAnnotFile (if provided)

sdrf (character, list or data.frame) optional extraction and adding of experimen- tal meta-data: if character, this may be the ID at ProteomeExchange. Besides, the output from readSdrf or a list from defineSamples may be provided; if gr is provided, it gets priority for grouping of replicates

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 ro absolute path), 1st is expected to corre- spond 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 (de- faults to low number of groups, i.e. higher number of samples per group)
titGraph (character) custom title to plot
wex (numeric) relative expansion factor of the violin in plot
plotGraph (logical) optional plot violplot 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

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 add-tional 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
readOpenMSFile

Examples

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

Description

Protein quantification results from 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 normalized 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 returning 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 - made as ‘species2’,’species3’,...); if list and list-element has multiple values they will be used for exact matching of accessions (ie 2nd of argument annotCol)

silent (logical) suppress messages

debug (logical) display additional messages for debugging

callFrom (character) allow easier tracking of message(s) produced

Details

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

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

Value

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

See Also

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

**Examples**

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

**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, too. Then relevant information gets extracted, the data can optionally normalized and displayed as boxplot or vioplot. The final output is a list containing 6 elements: `$raw`, `$quant`, `$annot`, `$counts`, `$quantNotes` and `$notes`. Alternatively, a data.frame with annotation and quantitation data may be returned if `separateAnnot=FALSE`. Note: There is no normalization by default since quite frequently data produced by Proline are already sufficiently normalized. The figure produced using the argument `plotGraph=TRUE` may help judging if the data appear sufficiently normalized (distributions should align).

**Usage**

```r
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,
  refl = NULL,
  separateAnnot = TRUE,
  plotGraph = TRUE,
  titGraph = NULL,
  wex = 2,
  specPref = c(conta = "_conta\|", mainSpecies = "OS=Homo sapiens"),
  gr = NULL,
  sdrf = NULL,
  supplAnnotFile = TRUE,
  groupPref = list(lowNumberOfGroups = TRUE),
  silent = 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 column names 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 column names 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
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 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)

`silent` (logical) suppress messages

`callFrom` (character) allow easier tracking of messages produced

`debug` (logical) display additional messages for debugging

Details

This function has been developed using Proline version 1.6.1 coupled with MS-Angel 1.6.1. The classical way of using 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 the 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

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

**Read Tabulated Files Exported By ProteomeDiscoverer At Protein Level**

**Description**

Depreciated 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**

```r
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`
readProtDiscovFile

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..PSMs","X..Unique.Peptides","X..AAs","MW..kDa."))
contamCol (character or integer, length=1) which columns should be used for contaminants marked by ProteomeDiscoverer. If a column named contamCol is found, the data will be lateron filtered to remove all contaminants, set to NULL for keeping all contaminants
refLi (character or integer) custom specify which line of data is main species, if character (eg 'mainSpe'), the column 'SpecType' in $annot will be searched for exact match of the (single) term given
separateAnnot (logical) if TRUE output will be organized as list with $annot, $abund for initial/raw abundance values and $quant with final 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 supplAnnotFile (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 futher 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
Details

This function has been replaced by readProteomeDiscovererFile (from the same package)! The syntax and structure of output has remained the same, you can simply replace the name of the function called.

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 lateron filtered to remove all contaminants, set to NULL for keeping all contaminants.

This function replaces the depreciated function readPDEexport.

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

```r
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)
```

---

**readProtDiscovPeptides**

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.
readProtDiscovPeptides

Usage

readProtDiscovPeptides(
  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"),
  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 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
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..PSMs","X..Unique.Peptides","X..AAs","MW..kDa.").)

contamCol (character or integer, length=1) which columns should be used for contaminants marked by ProteomeDiscoverer. If a column named contamCol is found, the data will be lateron filtered to remove all contaminants, set to NULL for keeping all contaminants

refLi (character or integer) custom specify which line of data is main species, if character (eg 'mainSpe'), the column 'SpecType' in $annot will be searched for exact match of the (single) term given

separateAnnot (logical) if TRUE output will be organized as list with $annot, $abund for initial/raw abundance values and $quant with final normalized quantitations

FDRCol (list) optional indication to search for protein FDR information

plotGraph (logical or integer) optional plot of type vioplot of initial and normalized data (using normalizeMeth); if integer, it will be passed to layout when plotting

titGraph (character) depreciated custom title to plot, please use 'tit'

wex (integer) relative expansion factor of the violin-plot (will be passed to vioplotW)

specPref (character or list) define characteristic text for recognizing (main) groups of species (1st for 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) 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 supAnnotFile 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 form peptide sequences and be displayed in $annot as columns 'prec' and 'foll'. If a column named contamCol is found, the data will be lateron filtered to remove all contaminants, set to NULL for keeping all contaminants This function replaces the depreciated function readPDExport.

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")

---

readProteomeDiscovererFile

Read Tabulated Files Exported By ProteomeDiscoverer At Protein LevelSM2

Description

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.

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),
  specPref = c(conta = "CON_|LYSC_CHICK", mainSpecies = "OS=Homo sapiens"),
  plotGraph = TRUE,
  wex = 1.6,
readProteomeDiscovererFile

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..AAs","MW..kDa.")
contamCol (character or integer, length=1) which columns should be used for contaminants marked by ProteomeDiscoverer. If a column named contamCol is found, the data will be lateron filtered to remove all contaminants, set to NULL for keeping all contaminants
refLi (character or integer) custom specify which line of data is main species, if character (eg 'mainSpe'), the column 'SpecType' in $annot will be searched for exact match of the (single) term given
separateAnnot (logical) if TRUE output will be organized as list with $annot, $abund for initial/raw abundance values and $quant with final 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 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 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)
**readProteomeDiscovererFile**

`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 - made 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 later on filtered to remove all contaminants, set to NULL for keeping all contaminants. This function replaces the depreciated function `readPDEexport`.

**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**

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

**Description**

Sample/experimental annotation meta-data form 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 adjusted automatically, if possible). This function returns a list with grouping of samples into replicates and additional information gathered. Input files compressed as .gz can be read as well.

**Usage**

```r
readSampleMetaData(
  quantMeth, 
  sdrf = NULL, 
  suplAnnotFile = NULL, 
  path = "." ,
  abund = NULL, 
  groupPref = list(lowNumberOfGroups = 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: if character, this may be the ID at ProteomeExchange or a similarly formatted local file. sdrf will get priority over suplAnnotFile, if provided.
- `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 in case of method="MQ" (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 to 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 method="PL" (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 lowNumberOfGroups=FALSE for automatically choosing a rather elevated number of groups if possible (defaults to low number of groups, i.e., higher number of samples per group)

silent (logical) suppress messages

debug (logical) additional messages for debugging
callFrom (character) allow 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 documented by the quantitation-software and/or from an sdrf repository and to attach it to the experimental data. This way, in subsequent steps of analysis (e.g., PCA, statistical tests) the user does not have to bother studying the experimental setup to figure out which samples should be considered as replicates 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 been implemented) 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 $lev 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 function returns an empty list.

See Also
this function is used internally by readMaxQuantFile, readProtDiscovFile etc; use readSdrf for reading sdrf-files, replicateStructure for mining annotation columns

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

This function allows reading proteomics meta-data from sdrf file, as they are provided on https://github.com/bigbio/proteomics-sample-metadata. 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

```r
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.

Value

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

See Also

in `read.table`
readUCSCtable

Examples

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

#### Description

This function allows reading and importing genomic UCSC-annotation data. Files can be read as default UCSC exprot or as GTF-format. In the context of proteomics we noticed that sometimes UniProt tables from UCSC are hard to match to identifiers from UniProt Fasta-files, i.e. 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, e.g. 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.

#### Usage

```r
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 reference 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

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

See Also
readUniProtExport

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

## results can be further combined with readUniProtExport()
delUniProtFi <- file.path(path1, "deUniProt_hg38chr11extr.tab")
delUniPr1 <- readUniProtExport(delUniProtFi, delUcsc=UcscAnnot1,
   targRegion="chr11:1-135,086,622")
delUniPr1[1:5,-5]

readUniProtExport 

Read protein annotation as exported from UniProt batch-conversion

Description
This function allows reading and importing protein-ID conversion results from UniProt. To do so,
first copy/paste your query IDs into UniProt 'Retrieve/ID mapping' field called '1. Provide your
identifiers' (or upload as file), verify '2. Select options'. In a typical case of 'enst000xxx' IDs you
may leave default settings, ie 'Ensemble Transcript' as input and 'UniProt KB' as output. Then,
'Submit' your search and 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.

Usage
readUniProtExport(
   UniProtFileNa,
   deUcsc = NULL,
   targRegion = NULL,
   useUniPrCol = NULL,
   silent = FALSE,
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

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 Ensemble 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`, `$strand`, `$Ensrnot`, `$avPos`)

See Also

- `readUCSCtable`

Examples

```r
path1 <- system.file("extdata",package="wrProteo")
deUniProtFi <- file.path(path1,"deUniProt_hg38chr11extr.tab")
deiUniPr1a <- readUniProtExport(deUniProtFi)
str(deiUniPr1a)

## 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 Read (Normalized) Quantitation Data Files Produced By Wombat At Protein Level

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, 
  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, 
)
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/

**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 fro '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 supplAnnotFile

**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
**readWombatNormFile**

- **gr** (character or factor) custom defined pattern of replicate association, will over-ride final grouping of replicates from sdrf and/or supplAnnotFile (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 futher 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 ro 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)

- **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.
removeSampleInList

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$quant)
matrixNAinspect(dataWB$quant, gr=gl(2,4))

Description

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

Usage

removeSampleInList(
  dat,
  remSamp,
  listElem = c("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) allow easier tracking of message(s) produced
Value

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

See Also

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)
Value

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

See Also

readMaxQuantFile, readProtDiscovFile, readProlineFile

Examples

dat <- list(quant=matrix(sample(11:99,9,replace=TRUE), ncol=3), annot=cbind(EntryName=c("YP0100_YEAST","",""),Accession=c("A5Z2X5","P01966","P35900"), SpecType=c("Yeast",NA,NA)))
replMissingProtNames(dat)

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",
  tit = NULL,
  color = 1,
  plotROC = TRUE,
  pch = 1,
  bg = NULL,
  overlPlot = FALSE,
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
summarizeForROC

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','lfdr' or 'p.value'), must be list-element of 'test'

thr (numeric) stat test (FDR/p-value) threshold, if NULL a panel of 108 p-value threshold-levels values will be used for calculating specificity and sensitivity

columnTest depreciated, 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

tit (character) optinal 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)

over1Plot (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 trouth' 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' designs 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 'alph') for statistical tests preformed prior to calling this function.

Note, that the choice of species-annotation plays a crucial role who the counting results are obtained. In case of multiple spike-in species the user should pay attention if they all are expected to change
test2grp

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 rocPKG Short, ROCr, pROC or ROCit

Value

This function returns a numeric matrix containing the columns ‘alph’, ‘spec’, ‘sens’, ‘prec’, ‘accur’, ‘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

test2grp 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,
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

```r
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[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)
```
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. 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.

Usage

```r
testRobustToNAimputation(
  dat,
  gr = NULL,
  annot = NULL,
  retnNA = TRUE,
  avSd = c(0.15, 0.5),
  avSdH = NULL,
  plotHist = FALSE,
  xLab = NULL,
  tit = NULL,
  imputMethod = "mode2",
  seedNo = NULL,
  multCorMeth = NULL,
  nLoop = 100,
  lfdrInclude = NULL,
  ROTSn = NULL,
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)
```
testRobustToNAimputation

Arguments

dat (matrix or data.frame) main data (may contain NA); if dat is list containing $quant and $annot as matrix, the element $quant will be used

gr (character or factor) replicate association; if dat contains a list-element $sampleSetup$groups or $sampleSetup$lev this may be used in case gr=NULL

annot (matrix or data.frame) annotation (lines must match lines of data !), if annot is NULL and argument dat is a list containing both $quant and $annot, the element $annot will be used

retnNA (logical) retain and report number of NA

avSd (numerical,length=2) population characteristics (mean and sd) for >1 NA-neighbours (per line)

avSdH depreciated, please use avSd instead; (numerical,length=2) population characteristics 'high' (mean and sd) for >1 NA-neighbours (per line)

plotHist (logical) additional histogram of original, imputed and resultant distribution (made using matrixNAneighbourImpute)

xLab (character) custom x-axis label

tit (character) custom title

imputMethod (character) choose the imputation method (may be 'mode2'(default), 'mode1', 'datQuant', 'modeAdopt', 'informed' or 'none', for details see matrixNAneighbourImpute)

seedNo (integer) seed-value for normal random values

multCorMeth (character) define which method(s) for correction of multiple testing should be run (for choice : 'BH','lfdr','BY','tValTab', choosing several is possible)

nLoop (integer) number of runs of independent NA-imputation

lfdrInclude (logical) depreciated, please used multCorMeth instead (include lfdr estimations, may cause warning message(s) concerning convergence if few too lines/proteins in dataset tested).

ROTSn (integer) depreciated, please used multCorMeth instead (number of repeats by ROTS, if NULL ROTS will not be called)

silent (logical) suppress messages

debug (logical) additional messages for debugging

callFrom (character) This function allows easier tracking of messages produced

Details

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
For details on choice of NA-imputation procedures with arguments 'imputMethod' and 'avSd' please see `matrixNAneighbourImpute`.

**Value**

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','lfdr' or 'ROTS.BH')

**See Also**

NA-imputation via `matrixNAneighbourImpute`, moderated t-test without NA-imputation `moderTest2grp`, calculating lfdr `pVal2lfdr`, eBayes in Bioconductor package `limma`, `t.test`, ROTS of Bioconductor package `ROTS`

**Examples**

```r
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[dat1<1] <- NA # mimick some NAs for low abundance
## normalize data
boxplot(dat1, main="data before normalization")
dat1 <- wrMisc::normalizeThis(as.matrix(dat1), meth="median")
## designate replicate relationships in samples ...
gr1 <- 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=70)
names(PLtestR1)
```

---

**Description**

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

**Usage**

```r
VolcanoPlotW2(
  Mvalue, # M-values
  pValue = NULL, # P-values
  useComp = 1, # Use conditional probability
  filtFin = NULL, # Filter for fineness
  ProjNa = NULL, # Project for NA
  FCthrs = NULL, # Fold changes
)
```
FdrList = NULL,
FdrThrs = NULL,
FdrType = NULL,
subTxt = NULL,
grayIncrem = TRUE,
col = NULL,
pch = 16,
compNa = NULL,
batchFig = FALSE,
cexMa = 1.8,
cexLa = 1.1,
limM = NULL,
limp = NULL,
annotColumn = 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())

filtFin (matrix or logical) The data may get filtered before plotting: If FALSE no filtering will get applied; if matrix of TRUE/FALSE it will be used as optional custom filter, otherwise (if Mvalue if an MArrayLM-object eg from limma) a default filtering based on the filtFin element will be applied

ProjNa (character) custom title

FCthrs (numeric) Fold-Change threshold (display as line) give as Fold-change and NOT log2(FC), default at 1.5, set to NA for omitting

FdrList (numeric) FDR data or name of list-element
FdrThrs (numeric) FDR threshold (display as line), default at 0.05, set to NA for omitting
FdrType (character) FDR-type to extract if Mvalue is 'MArrayLM'-object (eg produced by from moderTest2grp etc); if NULL it will search for suitable fields/values in this order : 'FDR','BH','lfdr' and 'BY'
subTxt (character) custom sub-title
grayIncrem (logical) if TRUE, display overlay of points as increased shades of gray
col (character) custom color(s) for points of plot (see also par)
pch (integer) type of symbol(s) to plot (default=16) (see also par)
compNa (character) names of groups compared
batchFig (logical) if TRUE figure title and axes legends will be kept shorter for display on fewer space
cexMa (numeric) font-size of title, as expansion factor (see also cex in par)
cexLa (numeric) size of axis-labels, as expansion factor (see also cex in par)
limM (numeric, length=2) range of axis M-values
limp (numeric, length=2) range of axis FDR / p-values
annotColumn (character) column names of annotation to be extracted (only if Mvalue is MArrayLM-object containing matrix $annot). The first entry (typically 'SpecType') is used for different symbols in figure, the second (typically 'GeneName') is used as preferred text for annotating the best points (if namesNBest allows to do so.)
annColor (character or integer) colors for specific groups of annotation (only if Mvalue is MArrayLM-object containing matrix $annot)
cexPt (numeric) size of points, as expansion factor (see also cex in par)
cexSub (numeric) size of subtitle, as expansion factor (see also cex in par)
cexTxLab (numeric) size of text-labels for points, as expansion factor (see also cex in par)
namesNBest (integer or character) number of best points to add names in figure; if 'passThr' all points passing FDR and FC-filtes will be selected; if the initial object Mvalue contains a list-element called 'annot' the second of the column specified in argument annotColumn will be used as text
NbestCol (character or integer) colors for text-labels of best points
sortLeg (character) sorting of 'SpecType' annotation either ascending ('ascend') or descending ('descend'), no sorting if NULL
NaSpecTypeAsContam (logical) consider lines/proteins with NA in Mvalue$annot[,"SpecType"] as contaminants (if a 'SpecType' for contaminants already exits)
useMar (numeric,length=4) custom margings (see also par)
returnData (logical) optional returning data.frame with (ID, Mvalue, pValue, FDRvalue, passFilt)
callFrom (character) allow easier tracking of message(s) produced
silent (logical) suppress messages
debug (logical) additional messages for debugging
**Value**

deprecated - returns nothing

**See Also**

this function was replaced by `plotPCAw`)

**Examples**

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

---

**Description**

Write sequences in fasta format to file

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

**Usage**

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

**Arguments**

- `prot` (character) vector of sequences, names will be used for fasta-header
- `fileNa` (character) name (and path) for file to be written
writeFasta2

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

**Details**

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

**Value**

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

**See Also**

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

**Examples**

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

.atomicMasses, 3
.checkKnitrProt, 4
.checkSetupGroups, 4
.commonSpecies, 5
.extrSpecPref, 6
.imputeNA, 7
.plotQuantDistr, 8
AAmass, 9, 14
AucROC, 10, 75

cleanListCoNames, 11
combineMultFilterN Aimpute, 12
convAASeq2mass, 14
convToNum, 10, 14, 24
corColumnOrder, 15
countNoOfCommonPeptides, 16

exportSdrfDraft, 17
extractTestingResults, 19
extrSpeciesAnnot, 20

foldChangeArrow, 21, 23
foldChangeArrow2, 21

grep, 12, 21

hist, 26

isolNAneighb, 23

lmFit, 20

MAplotW, 23
massDeFormula, 3, 10, 14, 24
matrixNAinspect, 25, 45, 71
matrixNAneighbourImpute, 8, 23, 26, 78, 79
moderTest2grp, 15, 19, 20, 29, 30, 73, 75, 76, 79
moderTestXgrp, 15, 16, 20

na.fail, 23, 26, 28

naOmit, 26
normalizeThis, 32, 34–36, 39–42, 44, 45, 47–50, 52, 54, 56, 57, 59–61, 69, 71

order, 16
par, 30, 74, 81
plotPCAw, 82
plotROC, 10, 29, 75
presenceFilt, 4, 12, 13, 31
pVal2lfdr, 76, 79

razorNoFilter, 30
read.table, 34, 36, 40, 42, 45, 48, 50, 53, 56, 59, 61, 64, 71
readDiaNNFile, 32
readDiaNNPeptides, 34
readFasta2, 17, 37, 83
readFragpipeFile, 5–7, 9, 38, 56, 61
readMassChroQFile, 40
readMaxQuantFile, 5–7, 9, 18, 34, 36, 40, 43, 48, 50, 56, 59, 61, 63, 73
readMaxQuantPeptides, 46
readOpenMSFile, 49
readProlineFile, 5–7, 9, 34, 36, 40, 42, 45, 48, 50, 51, 56, 61, 71, 73
readProtDiscovFile, 34, 36, 40, 50, 54, 63, 73
readProtDiscovPeptides, 56
readProteomeDiscovererFile, 45, 49, 59, 71
readSampleMetaData, 52
readSdrf, 63, 64
readUCSCTable, 65, 66, 67
readUniProtExport, 65, 66, 66
readWombatNormFile, 68
removeSampleInList, 71
replicateStructure, 63
replMissingProtNames, 72

scan, 38
stableMode, 23, 28
summarizeForROC, 10, 29, 30, 73

t.test, 75, 76, 79
test2grp, 74, 75, 75
testRobustToNAimputation, 23, 28, 72–75, 77

vioplotW, 33, 36, 39, 41, 50, 52, 55, 58, 61
VolcanoPlotW, 23
VolcanoPlotW2, 79

writeFasta2, 38, 82