Package ‘wrProteo’

January 8, 2020

Version 1.0.0

Title Proteomics Data Analysis Functions

Author Wolfgang Raffelsberger [aut, cre]

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

Description Data analysis of proteomics experiments by mass spectrometry is supported by this collection of functions mostly dedicated to the analysis of (bottom-up) quantitative (XIC) data. Fasta-formatted proteomes (eg from Uniprot) can be read with automatic parsing and multiple annotation types (like species origin, abbreviated gene names, etc) extracted. Quantitative proteomics measurements frequently contain multiple NA values, due to physical absence of given peptides in some samples, limitations in sensitivity or other reasons. The functions provided here help to inspect graphically the data to investigate the nature of NA-values via their respective replicate measurements and to help/confirm the choice of NA-replacement by low random values. Dedicated filtering and statistical testing using the framework of package 'limma' 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, can be run with special options consideration separating the data into sub-groups during normalization and testing. Subsequently, ROC curves can be constructed to compare multiple analysis approaches.

Depends R (>= 3.1.0)

Imports graphics, limma, stats, wrMisc

Suggests fdrtool, grDevices, MASS, RColorBrewer, ROTS, readxl

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2020-01-08 16:10:02 UTC
R topics documented:

`combineMultFilterNAimput` 2
`countNoOfCommonPeptides` 3
`extrSpeciesAnnot` 5
`matrixNAinspect` 6
`matrixNAneighbourImpute` 7
`plotROC` 8
`razorNoFilter` 10
`readFasta2` 11
`readProlineFile` 12
`removeSampleInList` 13
`summarizeForROC` 14
`test2grp` 15
`testRobustToNAimputation` 16

Index 18

---

`combineMultFilterNAimput`

*Combine multiple filters on NA-imputed data*

**Description**

In most omics data-analysis one needs to employ a certain number of filtering strategies to avoid getting artifacts to the step of statistical testing. `combineMultFilterNAimput` takes on one side the origial data and on the other side NA-imputed data to create several differnt filters and to finnally combine them. A filter aiming to take away the least abundant values (using the imputede data) is fine-tuned by the argument `abundThr`. This step compares the means for each group and line, at least one grou-mean has to be > the threshold (based on hypothesis that if all conditions represent 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 `colTotNa`, `minSpeNo` and `minTotNo` are used at this step. Basically, this step allows defining a minimum content of 'real' (ie non-NA) values for further considering the measurements as reliable. This part uses internally `presenceFilt` for filtering elevated content of NA per line. Finally, this function combines both filters (as matrix of FALSE and TRUE) on NA-imputed and original data and returs a vector of logical values if corresponding lines passe all filter criteria.

**Usage**

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

Compare in-silico digested proteomes for unique and shared peptides, counts per protein or as peptides. Compare in-silico digested proteomes for unique and shared peptides, counts per protein or as peptides. The in-silico digestion may be performed separately using the package [R](https://bioconductor.org/packages/release/bioc/html/cleaver.html#cleaver).

**Note:** input must be list (or multiple names lists) of proteins with their respective peptides (eg by in-silico digestion).

**Arguments**

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

**Value**

vector of logical values if corresponding line passes filter criteria

**See Also**

- `presenceFilt`

**Examples**

```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
datT6b <- matrixNAneighbourImpute(datT6,grp=gl(2,3))
datT6c <- combineMultFilterNAimput(datT6,datT6b,grp=gl(2,3),abundThr=2)
```
countNoOfCommonPeptides

Description

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

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

Usage

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

Arguments

- `...`: (list) multiple lists of (in-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.
- `callFrom`: (character) allows easier tracking of message(s) produced.

Value

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

See Also

`readFasta2` and/or `cleave-methods`

Examples

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

**Extract species annotation**

**Description**

`extrSpeciesAnnot` identifies species-related annotation (as suffix to identifiers) for data conning 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**

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

**Arguments**

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

**Value**

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

**See Also**

`grep`

**Examples**

```r
spec <- c("keratin_CONT", "AB_HUMAN", "CD_YEAST", "EF_G_HUMAN", "HI_HUMAN_ECOLI", "_YEAST_012")
extrSpeciesAnnot(spec)
```
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 measures containing NA-values should rather me omitted. Of course, such decisions do have a strong impact on further steps of data-analysis and should be performed with care.

Usage

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

Arguments

dat (matrix or data.frame) main numeric data

gr (charcter or factor) grouping of columns of dat indicating who is a replicate of whom (ie the length of ‘gr’ must be equivalent to the number of columns in ‘dat’)

retnNA (logical) report number of NAs in graphic

xLab (character) custom x-label

tit (character) custom title

xLim (numerical,length=2) custom x-axis limits

silent (logical) suppress messages

callFrom (character) allow easier tracking of messages produced

Value

graphic only

See Also

hist, na.fail, naOmit
matrixNAneighbourImpute

**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**

matrixNAneighbourImpute replaces NA-values based on group neighbours (based on grouping of columns in argument `gr`), assuming a Gaussian distribution. If a given element was measured twice (eg in two replicate measurements), 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 sub-population of these NA-neighbour values may be used to investigate the hypothesis that NA-values arose from very low signal. Indeed, in a number of experimental settings some actual measurements may not meet an arbitrary defined baseline (as 'zero') or may be too low to be distinguishable from noise that associated measures were initially recorded as NA. In several types of (quantitative) measurements in proteomics and transcriptomics this is known to happen. So this function allows to model and subsequently replace all NA-values by Gaussian random values based on the characteristics of NA-neighbours in the same data-set. However, defining these characteristics (via the arguments `avSdH` and `avSdL`) may be very delicate and visual verification of the plots produced is highly encouraged! If more than 300 NA-neighbours were detected, the imputation will be based on a more restricted sub-set of data with >1 NA values. Optionally a histogram may be plotted showing the initial, imputed and final distribution to check if the global hypothesis that NA-values arose from very low measurements and to appreciate the impact of the imputed values to the overall final distribution. Of course, all decisions to replace values do have a strong impact on further steps of data-analysis and should be performed with care.

**Usage**

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

**Arguments**

- **dat** (matrix or data.frame) main data (may contain NA)
- **gr** (character or factor) grouping of columns of 'dat', replicate association
- **retnNA** (logical) decide if NA values should be removed or retained
plotROC

Description

plotROC plots ROC curves based on results from `summarizeForROC`. Does not return any data, plot only.

Usage

```r
plotROC(dat, ..., useCol = 2:3, methNames = NULL, col = NULL,
        pch = 1, bg = NULL, tit = NULL, point05 = 0.05, pointSi = 0.85,
        nByMeth = NULL, colPanel = c(grDevices::grey(0.4), 2:5),
        speciesOrder = NULL, txtLoc = c(0.4, 0.3, 0.04), legCex = 0.72,
        silent = FALSE, callFrom = NULL)
```

Value

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

See Also

`hist`, `na.fail`, `naOmit`

Examples

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

- **dat** (matrix) from testing (e.g., `summarizeForROC`)
- **...** additional input (must be of same type of format as 'dat')
- **useCol** (integer or character) colors to be used
- **methNames** (character) names of methods (data-sets) to be displayed
- **col** (character) custom color
- **pch** (integer) type of symbol to be used (see `par`)
- **bg** (character) background color in plot (see `par`)
- **tit** (character) custom title
- **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
- **colPanel** (character) custom colors
- **speciesOrder** (integer) custom order of species in legend
- **txtLoc** (numeric) location for text
- **legCex** (numeric) `cex` expansion factor for legend
- **silent** (logical) suppress messages
- **callFrom** (character) allows easier tracking of messages produced

Value

- plot only

See Also

- `summarizeForROC`, `moderTest2grp`

Examples

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

Value

vector of logical values if corresponding line passes filter criteria

See Also

`presenceFilt`

Examples

```
set.seed(2019); datT <- matrix(sample.int(20,60,replace=TRUE),ncol=6, 
                               dimnames=list(letters[1:10],LETTERS[1:6])) -3
datT[,2] <- datT[,2] +2
datT[which(datT <0)] <- 0
razorNoFilter(datT,speNa="A",totNa="B")
```
readFasta2

Read file of protein sequences in fasta format. Read fasta formatted file (from Uniprot) to extract (protein) sequences and name. If tableOut=TRUE output may be organized as matrix for separating meta-annotation. Extract, see also \url{https://www.uniprot.org/help/fasta-headers}.

Description

Read file of protein sequences in fasta format. Read fasta formatted file (from Uniprot) to extract (protein) sequences and name. If tableOut=TRUE output may be organized as matrix for separating meta-annotation. Extract, see also \url{FASTA-headers}.

Usage

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

Arguments

filename (character) names fasta-file to be read
delim (character) delimiter at header-line
databaseSign (character) characters at beginning right after the ‘>’ (typically specifying the database-origin), they will be excluded from the sequence-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 \url{FASTA-headers}
cleanCols (logical) remove columns with all entries NA, if tableOut=TRUE
silent (logical) suppress messages
callFrom (character) allows easier tracking of message(s) produced
debug (logical) supplemental messages for debugging

Value

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

See Also

scan or \url{read.fasta}
Examples

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

Description

Quantification results form MS-Angel and Proline Proline should be first saved via Excel or LibreOffice as csv or tabulated txt. Such files can be read by this function and relevant information be extracted. The final output is a list containing 3 elements: $annot, $abund and optional $quant, or returns data.frame with entire content of file if separateAnnot=FALSE.

Usage

```r
readProlineFile(fileNa, wdir = NULL, logConvert = TRUE,
quantCol = "^abundance_*", annotCol = c("accession", "description",
"is_validated", "coverage", "X.sequences", "X.peptides",
"protein_set.score"), separateAnnot = TRUE, silent = FALSE,
callFrom = NULL)
```

Arguments

- `fileNa` (character) name of file to read
- `wdir` (character) optional path (note: Windows backslash should be protected or written as '/')
- `logConvert` (logical) convert numeric data as log2, will be placed in $quant
- `quantCol` (character) (character) exact col-names or if length=1 pattern to search among column-names for $quant
- `annotCol` (character) (character) exact col-names or if length=1 pattern to search among column-names for $annot
- `separateAnnot` (logical) separate annotation from numeric data (quantCol and annotCol must be defined)
- `silent` (logical) suppress messages
- `callFrom` (character) allow easier tracking of message produced

Value

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

See Also

read.table

Examples

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

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

Description

Remove samples/columns from list of matrixes

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

Usage

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

Arguments

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

Value

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

See Also

testRobustToNAimputation
summarizeForROC

**summarizeForROC**

Summarize statistical test result for plotting ROC-curves

### Description

summarizeForROC takes statistical testing results (obtained using `testRobustToNAimputation` or `moderTest2grp`, based on limma) and calculates specificity and sensitivity values for plotting ROC-curves along a panel of thresholds. Based on column from `test$annot` and argument 'spec' TP,FP,FN and TN are determined. See also [ROC on Wikipedia](https://en.wikipedia.org/wiki/Receiver_operating_characteristic) for explanations of TP,FP,FN and TN as well as examples. An optional plot may be produced, too. Return matrix with TP,FP,FN,TN,spec,sens,prec,accur and FDR count values along the various thresholds specified in column 'alph'.

### Usage

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

### Arguments

- **test** (class `MArrayLM`, S3-object from limma) from testing (eg `testRobustToNAimputation` or `test2grp`)
- **thr** (numeric) threshold, if NULL a panel of 108 values will be used for calculating specificity and sensitivity
- **tyThr** (character,length=1) type of test-result to be used for sensitivity and specificity calculations (eg 'BH','lfdr' or 'p.value'), must be list-element of 'test'
- **columnTest** (character or integer) only in case 'tyThr' is matrix (as typically the case after `testRobustToNAimputation`) : which column of 'test$tyThr' should be used as test-result
- **spec** (character) labels for species will be matched to column 'spec' of `test$annot` and used for sensitivity and specificity calculations. Important : 1st label for matrix (expected as constant) and subsequent labels for spike-ins (variable)
- **tit** (character) optional custom title in graph
- **color** (character or integer) color in graph
- **plotROC** (logical) toggle plot on or off
- **pch** (integer) type of symbol to be used (see `par`)
test2grp

Value

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

See Also
testRobustToNAimputation, moderTest2grp, test2grp, eBayes, t.test

Examples

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

test2grp t-test each line of 2 groups of data

Description

test2grp performs t-test on two groups of data using limma, this is a custom implementation of moderTest2grp for proteomics. The final object also includes the results without moderation by limma (e.g. 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, callFrom = NULL)

Arguments

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

Test robust to NA-imputation

description
testRobustToNAimputation replaces NA values based on group neighbours (based on grouping of columns in argument gr), following overall assumption of close to Gaussian distribution. Furthermore, it is assumed that NA-values originate from experimental settings where measurements at or below detection limit are 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. 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). Testing by package ROTS may optionally be included. This function returns limma-like S3 list-object further enriched by additional fields/elements.

usage
testRobustToNAimputation(dat, gr, annot = NULL, retnNA = TRUE,
   avSdH = c(0.18, 0.5), avSdL = c(0.1, 0.5), plotHist = FALSE,
   xLab = NULL, tit = NULL, seedNo = 2018, nLoop = 20,
   lfdrInclude = TRUE, ROTSn = NULL, silent = FALSE,
   callFrom = NULL)
Arguments

- **dat** (matrix or data.frame) main data (may contain NA)
- **gr** (character or factor) replicate association
- **annot** (matrix or data.frame) annotation (lines must match lines of data !)
- **retnNA** (logical) retain and report number of NA
- **avSdH** (numeric) population characteristics (mean and sd) for >1 NA neighbours 'high' (per line)
- **avSdL** (numeric) population characteristics (mean and sd) for >0 NA neighbours 'low' (per line)
- **plotHist** (logical) additional plot
- **xLab** (character) custom x-axis label
- **tit** (character) custom title
- **seedNo** (integer) seed-value for normal random values
- **nLoop** (integer) number of runs of independent NA-imputation
- **lfdrInclude** (logical) include lfdr estimations (may cause warning message(s) concerning convergence if few too lines/proteins in dataset tested).
- **ROTSn** (integer) number of repeats by ROTS, if NULL ROTS will not be called
- **silent** (logical) suppress messages
- **callFrom** (character) allows easier tracking of messages produced

Value

matrix including imputed values or list of final and matrix with number of imputed by group. Various options of multiple testing ccorrection are implemented ('BY','lfdr','p.value' and 'ROTS.BH'

See Also

- `moderTest2grp`, `pVal2lfdr`, `eBayes`, `t.test`, `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=gr1,retNA=TRUE,nLoop=100,ROTSn=0,lfdr=FALSE)
names(PLtestR1)
```
Index

combineMultFilterNAimput, 2
countNoOfCommonPeptides, 3

eBayes, 15, 17
eextrSpeciesAnnot, 5
grep, 5

hist, 6, 8

matrixNAinspect, 6
matrixNAneighbourImpute, 7
moderTest2grp, 9, 14–17

na.fail, 6, 8
naOmit, 6, 8

par, 9, 14, 15

plotROC, 8

presenceFilt, 2, 3, 10
pVal2lfdr, 16, 17

razorNoFilter, 10
read.fasta, 11
read.table, 13
readFasta2, 4, 11
readProlineFile, 12
removeSampleInList, 13
ROTS, 16, 17

scan, 11

summarizeForROC, 8, 9, 14

t.test, 15–17
test2grp, 14, 15, 15
testRobustToNAimputation, 13–15, 16