Package ‘wrMisc’

April 23, 2020

Version 1.2.0

Title Analyze Experimental High-Throughput (Omics) Data

Author Wolfgang Raffelsberger [aut, cre]

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

Description The efficient treatment and convenient analysis of experimental high-throughput (omics) data gets facilitated through this collection of diverse functions. Several functions address advanced object-conversions, like manipulating lists of lists or lists of arrays, reorganizing lists to arrays or into separate vectors, merging of multiple entries, etc. Another set of functions provides speed-optimized calculation of standard deviation (sd), coefficient of variance (CV) or standard error of the mean (SEM) for data in matrices or means per line with respect to additional grouping (eg n groups of replicates). Other functions facilitate dealing with non-redundant information, by indexing unique, adding counters to redundant or eliminating lines with respect redundancy in a given reference-column, etc. Help is provided to identify very closely matching numeric values to generate (partial) distance matrixes for very big data in a memory efficient manner or to reduce the complexity of large data-sets by combining very close values. Many times large experimental datasets need some additional filtering, adequate functions are provided. Batch reading (or writing) of sets of files and combining data to arrays is supported, too. Convenient data normalization is supported in various different modes, parameter estimation via permutations or boot-strap as well as flexible testing of multiple pair-wise combinations using the framework of 'limma' is provided, too.

VignetteBuilder knitr

Depends R (>= 3.1.0)

Imports grDevices, graphics, MASS, stats

Suggests BBmisc, boot, coin, data.tree, fdrtool, flexclust, knitr, limma, mixdist, NbClust, preprocessCore, RColorBrewer, rmarkdown, som, stringi, utils, VGAM, vsn, xlsx

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no
R topics documented:

addBeforeFileExtension ........................................... 5
adjBy2ptReg .......................................................... 5
arrayCV ............................................................... 6
asSepList ............................................................. 7
buildTree ............................................................. 8
cbindNR ............................................................... 9
ccheckAvSd ........................................................... 10
ccheckGrpOrder ..................................................... 11
ccheckGrpOrderSEM ................................................. 12
ccheckSimValueInSer ............................................... 13
ccheckStrictOrder .................................................... 14
ccheckVectLength ................................................... 15
cleanReplicates ...................................................... 15
closeMatchMatrix .................................................... 17
coinPermTest ......................................................... 18
colMedSds ............................................................. 19
colorAccording2 ...................................................... 20
colSds ................................................................. 21
combineByEitherFactor ............................................. 22
combineOverlapInfo ................................................ 23
combineRedBasedOnCol ............................................. 24
combineReplFromListToMatr ....................................... 25
combineSingleT ....................................................... 26
completeArrLst ....................................................... 26
ccontribToContigPerFrag ......................................... 27
cconv01toColNa ...................................................... 28
cconvColorToTransp ................................................. 29
cconvMatr2df ........................................................ 30
cconvToNum ........................................................... 31
correctToUnique ...................................................... 32
countCloseToLimits ................................................. 33
countSameStartEnd .................................................. 34
cutArrayInCluLike ................................................... 35
cutAtMultSites ....................................................... 36
cutToNgrp ............................................................. 36
diffCombin ........................................................... 37
diffPPM ................................................................. 38
elemCloseCoord ....................................................... 39
equLenNumber ......................................................... 40
exlExtrValues ......................................................... 40
exponNormalize ...................................................... 42
extr1chan .............................................................. 43
R topics documented:

extractLast2numericParts ........................................ 44
eextrColsDeX ...................................................... 45
eextrNumericFromMattr ............................................ 46
eextrSpcText ....................................................... 46
efilt3dimArr ......................................................... 47
filterList .......................................................... 48
filtSizeUniq ........................................................ 49
findCloseMatch ...................................................... 50
findRepeated ......................................................... 52
findSimilFrom2sets ................................................ 52
findUsableGroupRange ............................................. 54
firstLineOfDat ...................................................... 55
firstOfRepeated ..................................................... 55
firstOfRepLines ..................................................... 56
fuseAnnotMattr ..................................................... 57
fuseCommonListElem ............................................... 58
fusePairs ............................................................ 59
getListOfRepeatedByCol ........................................... 60
getValuesByUnique ................................................ 61
htmlSpecCharConv ................................................ 62
linRegrParamAndPVal .............................................. 63
listBatchReplace ................................................... 64
listGroupsByNames ................................................ 64
lmSelClu ............................................................. 65
lrbind ............................................................... 66
makeMAList ........................................................ 67
makeNRedMatr ....................................................... 68
matchNamesWithReverseParts ...................................... 69
matr2list ........................................................... 70
mergeSelCol ........................................................ 71
mergeSelCol3 ......................................................... 72
mergeW2 .............................................................. 73
minDiff .............................................................. 75
moderTest2grp ...................................................... 76
moderTestXgrp ....................................................... 77
naOmit ............................................................... 78
nFragments ........................................................ 79
nFragments0 ........................................................ 80
nNonNumChar ....................................................... 80
nonAmbiguousMat .................................................. 81
nonAmbiguousNum .................................................. 82
nonRedaDataFrame .................................................. 83
nonRedundLines ..................................................... 84
normalizeThis ....................................................... 84
organizeAsListOfRepl .............................................. 86
partialDist ........................................................ 87
partUnlist .......................................................... 88
pasteC ............................................................... 89
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>presenceFilt</td>
<td>89</td>
</tr>
<tr>
<td>pVal2fdr</td>
<td>90</td>
</tr>
<tr>
<td>randIndFx</td>
<td>91</td>
</tr>
<tr>
<td>rankToContigTab</td>
<td>92</td>
</tr>
<tr>
<td>ratioAllComb</td>
<td>93</td>
</tr>
<tr>
<td>ratioToPpm</td>
<td>94</td>
</tr>
<tr>
<td>readCsvBatch</td>
<td>95</td>
</tr>
<tr>
<td>readVarColumns</td>
<td>96</td>
</tr>
<tr>
<td>readXlsxBatch</td>
<td>98</td>
</tr>
<tr>
<td>reduceTable</td>
<td>99</td>
</tr>
<tr>
<td>regrBy1or2point</td>
<td>100</td>
</tr>
<tr>
<td>regrMultBy1or2point</td>
<td>101</td>
</tr>
<tr>
<td>renameColumns</td>
<td>102</td>
</tr>
<tr>
<td>replNAbyLow</td>
<td>103</td>
</tr>
<tr>
<td>replPlateCV</td>
<td>104</td>
</tr>
<tr>
<td>rmDupl2colMatr</td>
<td>105</td>
</tr>
<tr>
<td>rowCVs</td>
<td>106</td>
</tr>
<tr>
<td>rowGrpCV</td>
<td>106</td>
</tr>
<tr>
<td>rowGrpMeans</td>
<td>107</td>
</tr>
<tr>
<td>rowGrpSds</td>
<td>108</td>
</tr>
<tr>
<td>rowMedSds</td>
<td>109</td>
</tr>
<tr>
<td>rowSds</td>
<td>109</td>
</tr>
<tr>
<td>rowSEMs</td>
<td>110</td>
</tr>
<tr>
<td>searchDataPairs</td>
<td>111</td>
</tr>
<tr>
<td>searchLinesAtGivenSlope</td>
<td>112</td>
</tr>
<tr>
<td>simpleFragFig</td>
<td>113</td>
</tr>
<tr>
<td>singleLineAnova</td>
<td>114</td>
</tr>
<tr>
<td>sortBy2CategorAnd1IntCol</td>
<td>115</td>
</tr>
<tr>
<td>stableMode</td>
<td>116</td>
</tr>
<tr>
<td>stdErrMedBoot</td>
<td>117</td>
</tr>
<tr>
<td>summarizeCols</td>
<td>118</td>
</tr>
<tr>
<td>tableToPlot</td>
<td>119</td>
</tr>
<tr>
<td>test2factLimma</td>
<td>120</td>
</tr>
<tr>
<td>transpGraySca</td>
<td>121</td>
</tr>
<tr>
<td>treatTxtDuplicates</td>
<td>122</td>
</tr>
<tr>
<td>triCoord</td>
<td>123</td>
</tr>
<tr>
<td>tTestAllVal</td>
<td>124</td>
</tr>
<tr>
<td>uniqCountReport</td>
<td>125</td>
</tr>
<tr>
<td>upperMaCoord</td>
<td>126</td>
</tr>
<tr>
<td>withinRefRange</td>
<td>127</td>
</tr>
<tr>
<td>writeCsv</td>
<td>127</td>
</tr>
<tr>
<td>XYToDiffPpm</td>
<td>129</td>
</tr>
</tbody>
</table>

Index: 130
addBeforFileExtension  *Add text before file-extension*

### Description

This function helps changing character strings like file-names and allows adding the character vector 'add' (length 1) before the extension (defined by last '.') of the input string 'x'. Used for easily creating variants/additional filenames but keeping current extension.

### Usage

```r
addBeforFileExtension(x, add, sep = "_")
```

### Arguments

- **x**  
  main character vector

- **add**  
  character vector to be added

- **sep**  
  (character) separator between 'x' & 'add' (character, length 1)

### Value

modified character vector

### Examples

```r
addBeforFileExtension(c("abd.txt","ghg.ijij.txt","kjh"),"new")
```

---

adjBy2ptReg  *Linear rescaling of numeric vector or matrix*

### Description

adjBy2ptReg takes data within window defined by 'lims' and determines linear transformation so that these points get the regression characteristics 'regrTo', all other points (ie beyond the limits) will follow the same transformation. In other words, this function performs 'linear rescaling', by adjusting (normalizing) the vector 'dat' by linear regression so that points falling in 'lims' (list with upper & lower boundaries) will end up as 'regrTo'.

### Usage

```r
adjBy2ptReg(dat, lims, regrTo = c(0.1, 0.9), refLines = NULL, callFrom = NULL)
```
arrayCV

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>numeric vector, matrix or data.frame</td>
</tr>
<tr>
<td>lims</td>
<td>(list, length=2) should be list giving limits (list(lo=c(min,max),hi=c(min,max)) in data allowing identifying which points will be used for determining slope &amp; offset</td>
</tr>
<tr>
<td>regrTo</td>
<td>(numeric, length=2) to which characteristics data should be regressed</td>
</tr>
<tr>
<td>refLines</td>
<td>(NULL or integer) optional subselection of lines of dat (will be used internal as refDat)</td>
</tr>
<tr>
<td>callFrom</td>
<td>(character) for better tracking of use of functions</td>
</tr>
</tbody>
</table>

Value

matrix with normalized values

See Also

normalizeThis

Examples

```r
set.seed(2016); dat1 <- round(runif(50,0,100),1)
## extreme values will be further away :
adjBy2ptReg(dat1,lims=list(c(5,9),c(60,90)))
plot(dat1,adjBy2ptReg(dat1,lims=list(c(5,9),c(60,90))))
```

arrayCV

CV of array

Description

arrayCV gets CVs for replicates in 2 or 3 dim array and returns CVs as matrix. This function may be used to calculate CVs from replicate microtiter plates (eg 8x12) where replicates are typically done as multiple plates, ie initial matrixes that are the organized into arrays.

Usage

```r
arrayCV(arr, byDim = 3, silent = TRUE, callFrom = NULL)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arr</td>
<td>(3-dim) array of numeric data like where replicates are along one dimesion of the array</td>
</tr>
<tr>
<td>byDim</td>
<td>(integer) over which dimension replicates are found</td>
</tr>
<tr>
<td>silent</td>
<td>(logical) suppres messages</td>
</tr>
<tr>
<td>callFrom</td>
<td>(character) allow easier tracking of message produced</td>
</tr>
</tbody>
</table>
asSepList

Value
matrix of CV values

See Also
rowCVs, rowGrpCV, replPlateCV

Examples
set.seed(2016); dat1 <- matrix(c(runif(200)+rep(1:10,20)),ncol=10)
head(arrayCV(dat1,byDim=2))

---

asSepList

Organize data as separate list-entries

Description
asSepList allows reorganizing list into separate numeric vectors. For example, matrixes or data.frames
will be split into separate columns (differernt to partUnlist which maintains the original structure).
This function also works with lists of lists. This function may be helpful for reorganizing data for
plots.

Usage
asSepList(
y,  
asNumeric = TRUE,  
minLen = 4,  
fxArg = NULL,  
silent = FALSE,  
callFrom = NULL
)

Arguments
y list to be separated/split in vectors
asNumeric (logical) to transform all list-elements in simple numeric vectors (won’t work if
some entries are character)
minLen (integer) (currently use of this argument not implemeneted!) min length (or
number of rows), as add’l element to eliminate arguments given wo names when
asSepList is called in vioplot2
fxArg (character) optinal names to exclude if any (lazy matching) matches (to exclude
other arguments be misinterpreted as data, used in wrGraph::vioplot2)
silent (logical) suppress messages
callFrom (character) allow easier tracking of message(s) produced
Value

list, partially unlisted to vectors

See Also

gpartUnlist, unlist

Examples

bb <- list(fa=gl(2,2), c=3:33, L2=matrix(21:24,nc=2), li=list(li1=11:14, li2=data.frame(41:44)))
asSepList(bb)
lapply(bb,.asDF2)
partUnlist(lapply(bb,.asDF2))
**cbindNR**

**Value**

list with $paths (branches as matrix with columns 'sumLen' & 'n'), $usedNodes (character vector of all names used to build tree) and $tree (object from data.tree)

**See Also**

package data.tree original function used Node; in this package : for exploiting edge/tree related

issues simpleFragFig, countSameStartEnd and contribToContigPerFrag.

**Examples**

frag2 <- cbind(beg=c(2,3,7,13,15,7,9,7,3,7,5,7,3),end=c(6,12,8,20,20,19,12,12,4,12,7,12,4))
rownames(frag2) <- c("A","E","B","C","D","F","H","G","I","J","K","L","M","N")
buildTree(frag2)
countSameStartEnd(frag2)

cbindNR                 \hspace{1cm} \textit{cbind to non-redundant}

**Description**

cbindNR combines all matrixes given as arguments to non-redundant column names (by ADDING the number of 'duplicated' columns !). Thus, this function works similar to cbind, but allows combining multiple matrix-objects containing redundant column-names. Of course, all input-matrixes must have the same number of rows ! By default, the output gets sorted by column-names. Note, due to the use of '...' arguments must be given by their full argument-names, lazy evaluation might not recognize properly argument names.

**Usage**

cbindNR(
    
    ..., convertDFtoMatr = TRUE,
    sortOutput = TRUE,
    summarizeAs = "sum",
    silent = FALSE,
    callFrom = NULL
)

**Arguments**

... all matrixes to get combined in cbind way
convertDFtoMatr          (logical) decide if output should be converted to matrix
sortOutput              (logical) optional sorting by column-names
summarizeAs             (character) decide of combined values should get summed (default, 'sum') or averaged ('mean')
checkAvSd

silent (logical) suppress messages
callFrom (character) allow easier tracking of messages produced

Value

matrix or data.frame (as cbind would return)

See Also

cbind, nonAmbiguousNum, firstOfRepLines

Examples

ma1 <- matrix(1:6,ncol=3,dimnames=list(1:2,LETTERS[3:1]))
ma2 <- matrix(11:16,ncol=3,dimnames=list(1:2,LETTERS[3:5]))
cbindNR(ma1,ma2)
cbindNR(ma1,ma2,summarizeAs="mean")

checkAvSd checkAvSd
 Check how multiple groups of data separate or overlap based on mean
 +/- sd

Description

checkAvSd compares if/how neighbour groups separate/overlap via the 'engineering approach' (+/- 2 standard-deviations is similar to a=0.05 t.test). This approach may be used as less elegant alternative to (multi-group) logistic regression. The function uses 'daAv' as matrix of means (rows are tested for up/down character/progression) which get compared with boundaries taken from daSd (for Sd values of each mean in 'daAv').

Usage

checkAvSd(
  daAv,
daSd,
nByGr = NULL,
  multSd = 2,
  codeConst = "const",
  extSearch = FALSE,
  outAsLogical = TRUE,
  silent = FALSE,
callFrom = NULL
)
checkGrpOrder

Arguments

- `daAv` matrix or data.frame
- `daSd` matrix or data.frame
- `nByGr` optional specifying number of Elements per group, allows rather using SEM (adopt to variable n of different groups)
- `multSd` (numeric) the factor specifying how many sd values should be used as margin
- `codeConst` (character) which term/word to use when specifying 'constant'
- `extSearch` (logical) if TRUE, extend search to one group further (will call result 'nearUp' or 'nearDw')
- `outAsLogical` to switch between 2col-output (separate col for 'up' and 'down') or simple categorical vector ('const','okDw','okUp')
- `silent` (logical) suppress messages
- `callFrom` (character) allow easier tracking of message(s) produced

Value

vector describing character as 'const' or 'okUp','okDw' (or if extSearch=TRUE 'nearUp','nearDw')

See Also

rowGrpMeans

Examples

```r
mat1 <- matrix(rep(11:24,3)[1:40],byrow=TRUE,ncol=8)
checkGrpOrderSEM(mat1,grp=gl(3,3)[-1])
checkAvSd(rowGrpMeans(mat1,gl(3,3)[-1]),rowGrpSds(mat1,gl(3,3)[-1]) )
# consider variable n :
checkAvSd(rowGrpMeans(mat1,gl(3,3)[-1]),rowGrpSds(mat1,gl(3,3)[-1]),nByGr=c(2,3,3))
```

Description

checkGrpOrder tests each line of 'x' if expected order appears. Used for comparing groups of measures with expected profile (simply by matching expected order)

Usage

```r
checkGrpOrder(x, rankExp = NULL, revRank = TRUE)
```
checkGrpOrderSEM

Arguments

x         matrix or data.frame
rankExp   (numeric) expected order for values in columns, default 'rankExp' =1:ncol(x)
revRank   (logical) if 'revRank'=TRUE, the initial ranks & reversed ranks will be tested

Value

vector of logical values

See Also

checkGrpOrderSEM

Examples

set.seed(2005); mat <- matrix(round(runif(40),1),ncol=4)
checkGrpOrder(mat)
checkGrpOrder(mat,c(1,4,3,2))

checkGrpOrderSEM       Check order of multiple groups including non-overlapping SEM-margins

Description

checkGrpOrderSEM tests each line of 'x' if expected order of (replicate-) groups (defined in 'grp') appears intact, while including SEM of groups (replicates) via a proportional weight 'sdFact' as (avGr1-gr1SEM) < (avGr1+gr1SEM) < (avGr2-gr2SEM) < (avGr2+gr2SEM). Used for comparing groups of measures with expected profile (by matching expected order) to check if data in 'x' representing groups ('grp') as lines follow. Groups of size=1: The sd (and SEM) can’t be estimated directly without any replicates, however, an estimate can be given by shrinking if 'shrink1sampSd'=TRUE under the hypothesis that the overall mechanisms determining the variances is constant across all samples.

Usage

checkGrpOrderSEM(
  x,
  grp,
  sdFact = 1,
  revRank = TRUE,
  shrink1sampSd = TRUE,
  silent = FALSE,
  callFrom = NULL
)
checkSimValueInSer

Arguments

- **x**: matrix or data.frame
- **grp**: (factor) to organize replicate columns of (x)
- **sdFact**: (numeric) is proportional factor how many units of SEM will be used for defining lower & upper bounds of each group
- **revRank**: (logical) optionally revert ranks
- **shrink1sampSd**: (logical)
- **silent**: (logical) suppress messages
- **callFrom**: (character) allow easier tracking of message(s) produced

Value

logical vector if order correct (as expected based on ranks)

See Also

takes only 10

Examples

```r
mat1 <- matrix(rep(11:24,3)[1:40],byrow=TRUE,ncol=8)
checkGrpOrderSEM(mat1,grp=gl(3,3)[-1])
```

---

checkSimValueInSer **Check for similar values in series**

Description

checkSimValueInSer checks all values of 'x' for similar values outside/within (relative) range of 'ppm' (ie ambiguous within given range). Return logical vector: FALSE for each entry of 'x' if value inside of ppm range to neighbour

Usage

checkSimValueInSer(x, ppm = 5, sortX = TRUE)

Arguments

- **x**: numeric vector
- **ppm**: (numeric) ppm-range for considering as similar
- **sortX**: (logical) allows speeding up function when set to FALSE, for large data that are already sorted

Value

logical vector: FALSE for each entry of 'x' if value inside of ppm range to neighbour
checkStrictOrder

Check for strict (ascending or descending) order

Description
checkStrictOrder tests lines of 'dat' (matrix of data.frame) for strict order (ascending, descending or constant), each col of data is tested relative to the col on its left.

Usage
checkStrictOrder(dat, invertCount = TRUE)

Arguments
dat     matrix or data.frame
invertCount     (logical)

Value
matrix with counts of (non-)up pairs, (non-)down pairs, (non-)equal-pairs, if 'invertCount'=TRUE resulting 0 means that all columns are following the described characteristics (with variabale col-numbers easier to count)

See Also
order

Examples
set.seed(2005); mat <- matrix(round(runif(40),1),nc=4)
checkStrictOrder(mat); mat[which(checkStrictOrder(mat)[,2]==0),]
checkVectLength

Description

checkVectLength checks argument 'x' for expected length 'expeL' and return either message or error when expectation not met. Used for parameter ('sanity') checking in other user front-end functions.

Usage

checkVectLength(
  x,
  expeL = 1,
  stopOnProblem = FALSE,
  silent = FALSE,
  callFrom = NULL
)

Arguments

x (numeric or character vector) input to check length
expeL (numeric) expected length
stopOnProblem (logical) continue on problems with message or stop (as error message)
silent (logical) suppress messages if TRUE
callFrom (character) allows easier tracking of message(s) produced

Value

NULL (produces only optional message if length is OK or error-message if length is not OK)

Examples

aa <- 1:5; checkVectLength(aa, exp=3)

---
cleanReplicates

Replace most distant values by NA
Description

This procedures aims to strengthen (clean) the most extreme of noisy replicates by identifying the most distant points (among a set of replicates). The input 'x' (matrix or data.frame) is supposed to come from multiple different measures taken in replicates (eg weight of different individuals as rows taken as multiple replicate measures in subsequent columns). With the argument nOut1 the user chooses the total number of most extreme values to replace by NA. how many of the most extreme replicates of the whole dataset will replaced by NA, ie with nOut1=1 only the single most extreme outlyer will be replaced by NA. Outlier points are determined as point(s) with highest distance to (row) center (median and mean choice via argument 'centrMeth'). Returns input data with "removed" points set to NA, or if retOffPos=TRUE the most extreme/outlier positions.

Usage

cleanReplicates(
x, 
  centrMeth = "median", 
  nOut1 = 2, 
  retOffPos = FALSE, 
  silent = FALSE, 
  callFrom = NULL 
)

Arguments

x matrix (or data.frame)
centrMeth (character) method to summarize (mean or median)
nOut1 (integer) determines how many points per line will be set to NA (with n=1 the worst row of replicates will be 'cleaned')
retOffPos (logical) if TRUE, replace the most extreme outlyer only
silent (logical) suppres messages
callFrom (character) allow easier tracking of message(s) produced

Value

input data with "removed" points set as NA, or if retOffPos=TRUE the most extreme/outlier positions

Examples

mat3 <- matrix(c(19,20,30, 18,19,28, 16,14,35),ncol=3)
cleanReplicates(mat3,nOut1=1)
closeMatchMatrix

Reorganize results of search for close (similar) values in matrix-view

Description

closeMatchMatrix reorganizes/refines results from simple search of similar values of 2 sets of data by findCloseMatch (as list for one-to many relations) to more human friendly/readable matrix. This function returns results combining two sets of data which were initially compared (eg measured and theoretical values) as matrix-view using output of findCloseMatch and both original datasets. Additional information (covariables, annotation, ...) may be included as optional columns for either 'predMatr' or 'measMatr'. Note: It is important to run findCloseMatch with sortMatch=FALSE! Note: Results presented based on view of 'predMatr', so if multiple 'measMatr' are at within tolered distance, lines of 'measMatr' will be repeated; Note: Distances 'disToMeas' and 'ppmToPred' are oriented: neg value if measured is lower than predicted (and pos values if higher than predicted); Note: Returns NULL when nothing within given limits of comparison;

Usage

closeMatchMatrix(
  closeMatch,
  predMatr,
  measMatr,
  prefMatch = c("^x", "^y"),
  colPred = 1,
  colMeas = 1,
  limitToBest = TRUE,
  asDataFrame = FALSE,
  origNa = TRUE,
  silent = FALSE,
  callFrom = NULL,
  debug = FALSE
)

Arguments

closeMatch (list) output from findCloseMatch, ie list with hits for each 'x' (1st argument) : named vectors of value & x index in name; run with 'sortMatch'=F

predMatr (vector or matrix) predicted values, the column 'colPred' indicates which column is used for matching from findCloseMatch; if column 'id' present this column will be used as identifier for matching

measMatr (vector or matrix) measured values, the column 'colMeas' indicates which column is used for matching from findCloseMatch; if column 'id' present this column will be used as identifier for matching

prefMatch (character, length=2) prefixes ('^x' and/or '^y') that may have been added by findCloseMatch

colPred (integer or text, length=1) column of 'predMatr' with main values of comparison
colMeas (integer or text, length=1) column of 'measMatr' with main measures of comparison
limitToBest (integer) column of 'measMatr' with main measures of comparison
asDataFrame (logical) convert results to data.frame if non-numeric matrix produced (may slightly slow down big results)
origNa (logical) will try to use original names of objects 'predMatr','measMatr', if they are not multi-column and not conflicting other output-names (otherwise 'predMatr','measMatr' will appear)
silent (logical) suppress messages
callFrom (character) allows easier tracking of message(s) produced
debug (logical) for bug-tracking: more/enhanced messages

Value
results as matrix-view based on initial results from findCloseMatch, including optional columns of supplelemental data for both sets of data for comparison. Returns NULL when nothing within limits

See Also
findCloseMatch, checkSimValueInSer

Examples
aA <- c(11:17); bB <- c(12.001,13.999); cC <- c(16.2,8,9,12.5,15.9,13.5,15.7,14.1,5)
(cloMa <- findCloseMatch(aA,cC,com="diff",lim=0.5,sor=FALSE))
# all matches (of 2d arg) to/within limit for each of 1st arg ('x'); 'y' ..to 2nd arg = cC
(maAa <- closeMatchMatrix(cloMa,aA,cC,lim=TRUE)) #
(maAa <- closeMatchMatrix(cloMa,aA,cC,lim=FALSE,origN=TRUE)) #
(maAa <- closeMatchMatrix(cloMa,cbind(valA=81:87,aA),cbind(valC=91:99,cC),colM=2,
colP=2,lim=FALSE))
(maAa <- closeMatchMatrix(cloMa,cbind(aA,valA=81:87),cC,lim=FALSE,deb=TRUE)) #
a2 <- aA; names(a2) <- letters[1:length(a2)]; c2 <- cC; names(c2) <- letters[10+1:length(c2)]
(cloM2 <- findCloseMatch(x=a2,y=c2,com="diff",lim=0.5,sor=FALSE))
(maA2 <- closeMatchMatrix(cloM2,predM=cbind(valA=81:87,a2),measM=cbind(valC=91:99,c2),
colM=2,colP=2,lim=FALSE,asData=TRUE))
(maA2 <- closeMatchMatrix(cloM2,cbind(id=names(a2),valA=81:87,a2),cbind(id=names(c2),
valC=91:99,c2),colM=3,colP=3,lim=FALSE,deb=FALSE))

coinPermTest

Compare means of two vectors by permutation test

Description
Run coin-flipping like permutation tests (to compare difference of 2 means: 'x1' and 'x2') without any distribution-assumptions. Uses the package coin.
Usage

    coinPermTest(
        x1,
        x2,
        orient = "two.sided",
        nPerm = 5000,
        silent = FALSE,
        callFrom = NULL
    )

Arguments

x1  numeric vector (to be compared with vector 'x2')
x2  numeric vector (to be compared with vector 'x1')
orient  (character) may be "two.sided","greater" or "less"
nPerm  (integer) number of permutations
silent  (logical) suppress messages
callFrom  (character) allow easier tracking of message(s) produced

Value

"MCp" class numeric output with p-values

See Also

oneway_test in LocationTests

Examples

    coinPermTest(2,3)

---

**colMedSds**  

*Standard error of median for each column by bootstrap*

Description

Determine standard error (sd) of median by bootstraping for multiple sets of data (rows in input matrix 'dat'). Note: Uses the package boot.

Usage

    colMedSds(dat, nBoot = 99)

Arguments

dat  (numeric) matix
nBoot  (integer) number if iterations
colorAccording2

Value

(character) color input

gradTy

(character) type of gradient may be 'rainbow', 'heat.colors', 'terrain.colors', 'cm.colors', 'grey.colors', 'gray.colorsW' or 'logGray'

nStartOmit

(integer) omit n steps from beginning of gradient range

nEndOmit

(integer) omit n steps from end of gradient range

revCol

(logical) reverse order

alpha

(numeric) optional transparency value (1 for no transparency, 0 for complete opaqueness)

callFrom

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

Value

(character) color input

Examples

set.seed(2016); dat1 <- matrix(c(runif(200)+rep(1:10,20)),ncol=10)
colMedSds(dat1)

References:

This function helps making color-gradients for plotting a numerical variable.
colSds

See Also

cut

Examples

set.seed(2015); dat1 <- round(runif(15),2)
plot(1:15,d1,pch=16,cex=2,col=colorAccording2(dat1))
plot(1:15,d1,pch=16,cex=2,col=colorAccording2(dat1,nStart0=0,nEnd0=4,revCol=TRUE))
plot(1:9,pch=3)
points(1:9,1:9, col=transpGraySca(st=0, en=0.8, nSt=9, trans=0.3), cex=42, pch=16)

colSds

sd for each column

Description

colSds is a speed optimized sd for matrix or data.frames. It and treats each line as an independent set of data for calculating the sd (equiv to apply(dat,1,sd)). NAs are ignored from data.

Usage

colSds(dat)

Arguments

dat matrix (or data.frame) with numeric values (may contain NAs)

Value

numeric vector of sd values

See Also

sd

Examples

set.seed(2016); dat1 <- matrix(c(runif(200)+rep(1:10,20)),nc=10)
colSds(dat1)
combineByEitherFactor  Create factor-like column regrouping data regrouping simultaneously by two factors

Description
This function aims to address the situation when two somehow different groupings (of the same data) exist and need to be joined. It is not necessary that both alternative groupings use the same labels, neither. combineByEitherFactor adds new (last) column named 'grp' to input matrix representing the combined factor relative to 2 specified columns from input matrix 'mat' (via 'refC1','refC2'). Optionally, the output may be sorted and a column giving n per factor-level may be added. The function treats selected columns of 'mat' as pairwise combination of 2 elements (that may occur multiple times over all lines of 'mat') and sorts/organizes all instances of such combined elements (ie from both selected columns) as repeats of a given group, who's class number is given in output column 'grp', the (total) number of repeats may be displayed in column 'nGrp' (nByGrp=TRUE). If groups are overlapping (after re-ordering), an iterative process of max 3x2 passes will be launched after initial matching. Works on numeric as well as character input.

Usage

combineByEitherFactor(
  mat,
  refC1,
  refC2,
  nByGrp = FALSE,
  convergeMax = TRUE,
  callFrom = NULL,
  silent = FALSE
)

Arguments

mat    input matrix
refC1  (numeric) column-number of 'mat' to use as 1st set
refC2  (numeric) column-number of 'mat' to use as 2nd set
nByGrp (logical) add last col with n by group
convergeMax (logical) if TRUE, run 2 add’l iterative steps to search convergence to stable result
callFrom (character) allows easier tracking of message(s) produced
silent  (logical) suppres messages

Value
matrix containing both selected columns plus additional column(s) indicating group-number of the pair-wise combination (and optional the total n by group)
combineOverlapInfo

**Examples**

```r
nn <- rep(c("a","e","b","c","d","g","f"),c(3,1,2,1,2,1,2))
qq <- rep(c("m","n","p","o","q"),c(2,1,1,4,4))
nq <- cbind(nn,qq)[c(4,2,9,11,6,10,7,3,5,1,12,8),]
combineByEitherFactor(nq,1,2,nBy=TRUE); combineByEitherFactor(nq,1,2,nBy=FALSE)
combineByEitherFactor(nq,1,2,conv=FALSE); combineByEitherFactor(nq,1,2,conv=TRUE)
```

```r
mm <- rep(c("a","b","c","d","e"),c(3,4,2,3,1)); pp <- rep(c("m","n","o","p","q"),c(2,2,2,2,5))
combineByEitherFactor(cbind(mm,pp),1,2,con=FALSE,nBy=TRUE);
combineByEitherFactor(cbind(mm,pp),1,2,con=TRUE,nBy=TRUE)
```

**Description**

Search points in x,y space that are located very close and thus likely to overlap. In case of points close enough, various options for joining names (and shortening longer descriptions) are available.

**Usage**

```r
combineOverlapInfo(
  dat,
  suplInfo = NULL,
  disThr = 0.01,
  addNsimil = TRUE,
  txtSepChar = ",",
  combSym = "+",
  maxOverl = 50,
  callFrom = NULL,
  debug = FALSE,
  silent = FALSE
)
```

**Arguments**

- **dat** (matrix) matrix or data.frame with 2 cols (used ONLY 1st & 2nd column !), used as x & y coordinates
- **suplInfo** (NULL or character) when points are considered overlapping the text from 'suplInfo' will be reduced to fragment before 'txtSepChar' and combined (with others from overlapping text) using 'combSym', if NULL $combInf will appear with row-numbers
- **disThr** (numeric) distance-threshold for considering as similar via searchDataPairs()
- **addNsimil** (logical) include number of fused points
- **txtSepChar** (character) for use with retain1stPart(): where to cut (& keep 1st part) text from 'suplInfo' to return in out$CombInf; only 1st element used!
combineRedBasedOnCol

parameter:

combSym (character) concatenation symbol (character, length=1) for points considered overlying, see also ‘suplInfo’

maxOverl (integer) if NULL no limit or max limit of group/clu size (avoid condensing too many neighbour points to single cloud)

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

debug (logical) additional messages for debugging

silent (logical) suppres messages

Value

matrix with fused (condensed) information for cluster of overlapping points

Examples

set.seed(2013)
datT2 <- matrix(round(rnorm(200)+3,1),ncol=2,dimnames=list(paste("li",1:100,sep=""), letters[23:24]))

# (mimick) some short and longer names for each line
inf2 <- cbind(sh=paste(rep(letters[1:4],each=26),rep(letters,4),1:(26*4),sep=""),
lo=paste(rep(LETTERS[1:4],each=26),rep(LETTERS,4),1:(26*4),",",rep(letters[sample.int(26)],4),
rep(letters[sample.int(26)],4),sep=""))[1:100,]

head(datT2,n=10)
head(combineOverlapInfo(datT2,disThr=0.03),n=10)
head(combineOverlapInfo(datT2,suplI=inf2[,2],disThr=0.03),n=10)

Description

This function works similar to unique, but it takes a matrix as input and considers one specified column to find unique instances. It identifies 'repeated' lines of the input-matrix (or data.frame) 'mat' based on (repeated) elements in/of column with name 'colNa' (or column-number). Redundant lines (ie repeated lines) will disappear in output. Eg used with extracted annotation where 1 gene has many lines for different GO annotation.

Usage

combineRedBasedOnCol(mat, colNa, sep = ",", silent = FALSE, callFrom = NULL)

Arguments

mat input matrix or data.frame

colNa character vector (length 1) macting 1 column name (if mult only 1st will be used), in case of mult matches only 1st used

sep (character) separator (default=",")

silent (logical) suppress messages

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

matrix containing the input matrix without lines considered repeated (unique-like)

**See Also**

findRepeated, firstOfRepLines, organizeAsListOfRepl

**Examples**

```r
matr <- matrix(c(letters[1:6], "h", "h", "f", "e", LETTERS[1:5]), ncol=3,
      dimnames=list(letters[11:15], c("xA", "xB", "xC")))
combineRedBasedOnCol(matr, colN="xB")
combineRedBasedOnCol(rbind(matr[1,], matr), colN="xB")
```

---

**Description**

Suppose multiple measures (like multiple channels) are taken for subjects and these measures are organized as groups in a list, like multiple parameters (= channels) or types of measurements (typically many parameters are recorded when screening compounds in microtiter plates). Within one parameter/channel all replicate-data from separate list-entries ("lst") will get combined according to names of list-elements. The function will trim any redundant text in names of list-elements, try to isolate separator (may vary among replicate-groups, but should be 1 character long). eg names "hct116 1.1.xlsx" & "hct116 1.2.xlsx" will be combined as replicates, "hct116 2.1.xlsx" will be considered as new group.

**Usage**

```r
combineReplFromListToMatr(lst, callFrom = NULL)
```

**Arguments**

- `lst` (list) list of arrays (typically multi-parameter measures of microtiterplate data)
- `callFrom` (character) allows easier tracking of message(s) produced

**Value**

list of arrays now with same dimension of arrays (but shorter, since replicate-arrays were combined)

**See Also**

extr1chan, organizeAsListOfRepl
Examples

```r
lst2 <- list(aa_1x=matrix(1:12,nrow=4,byrow=TRUE),ab_2x=matrix(24:13,nrow=4,byrow=TRUE))
combineReplFromListToMatr(lst2)
```

**completeArrLst**

Complete list of arrays for same dimensions

Description

This function aims to inspect repeating structures of data given as list of arrays and will try to complete arrays with fewer lines or columns (as this may appear eg with the very last set of high-throughput screening data if fewer measures remain in the last set). Thus, the dimensions of the arrays are compared and cases with fewer (lost) columns (eg fewer experimental replicates) will be adjust/complete by adding column(s) of NA. Used eg when at reading microtiterplate data the last set is not complete.
contribToContigPerFrag

Usage

    completeArrLst(arrLst, silent = FALSE, callFrom = NULL)

Arguments

arrLst  (list) list of arrays (typically 1st and 2nd dim for specific genes/objects, 3rd for different measures associated with)
silent  (logical) suppress messages
callFrom (character) allows easier tracking of message(s) produced

Value

list of arrays, now with same dimension of arrays

See Also

organizeAsListOfRepl, extr1chan

Examples

    arr1 <- array(1:24,dim=c(4,3,2),dimnames=list(c(LETTERS[1:4]),
                                   paste("col",1:3,sep=""),c("ch1","ch2")))
    arr3 <- array(81:96,dim=c(4,2,2),dimnames=list(c(LETTERS[1:4]),
                                   paste("col",1:2,sep=""),c("ch1","ch2")))
    arrL3 <- list(pl1=arr1,pl3=arr3)
    completeArrLst(arrL3)

contribToContigPerFrag

Characterize individual contribution of single edges in tree-structures

Description

This function helps investigating tree-like structures with the aim of indicating how much individual tree components contribute to compose long stretches. contribToContigPerFrag characterizes individual (isolated) contribution of single edges in tree-structures. Typically used to process/exploit summarized trees (as matrix) made by buildTree which makes use of the package data.tree. For example if A,B and C can be joined as well and B +D, this function will check if A+B+C is longer and if A contributes to the longest tree.

Usage

    contribToContigPerFrag(joinMat, fullLength = NULL, nDig = 3)
Arguments

joinMat (matrix) matrix with concatenated edges as rownames (separated by slashes), column sumLen for total length and column n for number of edges

fullLength (integer) custom total length (useful if the concatenated edges do not cover 100 percent of the original precursor whose fragments are studied)

nDig (integer) rounding: number of digits for 3rd column len.rat in output

Value

matrix of 3 columns: with length of longest tree-branches where given edge participates (column sumLen), the (total) number of edges therein (col n.frag) and a relative value (len.rat)

See Also
to build tree buildTree

Examples

path1 <- matrix(c(17,19,18,17, 4,4,2,3),ncol=2, dimnames=list(c("A/B/C/D","A/B/G/D","A/H","A/H/I"),c("sumLen","n")))
contribToContigPerFrag(path1)

conv01toColNa Convert matrix of integer to matrix of x-times repeated column-names

Description

conv01toColNa transforms matrix of integers (eg 0 and 1) to repeated & concatenated text from 'colNa', the character string for 0 occurrances 'zeroTex' may be customized. Used eg when specifying (and concatenating) various counted elements (eg properties) along a vector like variable peptide modifications in proteomics.

Usage

conv01toColNa(mat, colNa = NULL, zeroTex = "", pasteCol = FALSE)

Arguments

mat input matrix (with integer values)

colNa alternative (column-)names to the ones from 'mat' (default colnames of mat)

zeroTex text to display if 0 (default ")"

pasteCol (logical) allows to collapse all columns to single chain of characters in output

Value

character vector
**Examples**

(ma1 <- matrix(sample(0:3,40,repl=TRUE),ncol=4,dimnames=list(NULL,letters[11:14])))
conv01toColNa(ma1)
conv01toColNa(ma1,colNa=LETTERS[1:4],ze=".")
conv01toColNa(ma1,colNa=LETTERS[1:4],pasteCol=TRUE)

---

**convColorToTransp**  
*Assign new transparency to given colors*

**Description**

This function allows (re-)defining a new transparency. A color encoding vector will be transformed to the same color(s) but with new transparency (alpha).

**Usage**

`convColorToTransp(color, alph = 1)`

**Arguments**

- `color` (character) color input
- `alph` (numeric) transparency value (1 for no transparency, 0 for complete opacity), values <1 will be treated as percent-values

**Value**

character vector (of same length as input) with color encoding for new transparency

**See Also**

`rgb`, `par`

**Examples**

col0 <- c("#998FCC","#5AC3BA","#CBD34E","#FF7D73")
col1 <- convColorToTransp(col0,alph=0.7)
layout(1:2)
pie(rep(1,length(col0)),col=col0)
pie(rep(1,length(col1)),col=col1,main="new transparency")
convMatr2df

Convert matrix (eg with redundant) row-names to data.frame

Description

convMatr2df provides flexible converting of matrix to data.frame. For example repeated/redundant rownames are not allowed in data.frame(), thus the corresponding column-names have to be renamed using a counter-suffix. In case of non-redundant rownames, a new column ‘addIniNa’ will be introduced at beginning to document the initial (redundant) rownames, non-redundant rownames will be created. Finally, this function converts the corrected matrix to data.frame and checks/converts columns for transforming character to numeric. If the input is a data.frame containing factors, they will be converted to character before potential conversion. Note: for simpler version (only text to numeric) see from this package .convertMatrToNum.

Usage

convMatr2df(
  mat,
  addIniNa = TRUE,
  duplTxtSep = "_",
  silent = FALSE,
  callFrom = NULL
)

Arguments

mat matrix (or data.frame) to be converted
addIniNa (logical) if TRUE an additional column ('ID') with rownames will be added at beginning
duplTxtSep (character) separator for enumerating replicated names
silent (logical) suppress messages
callFrom (character) allow easier tracking of message(s) produced

Value
data.frame

See Also

for simpler version (only text to numeric) see from this package .convertMatrToNum

Examples

dat1 <- matrix(1:10,ncol=2)
rownames(dat1) <- letters[c(1:3,2,5)]
## as.data.frame(dat1) ... would result in an error
convMatr2df(dat1)
convToNum checks if input vector/character string contains numbers (with or without comma) and attempts converting to numeric. This function was designed for extracting the numeric part of character-vectors (or matrix) containing both numbers and character-elements. Depending on the parameters convert and remove text-entries can be converted to NA (in resulting numeric objects) or removed (the number of elements/lines gets reduced, in consequence). Note: if 'x' is a matrix, its matrix-dimensions & -names will be preserved. Note: so far Inf and -Inf do not get recognized as numeric.

Usage

convToNum(
  x,
  spaceRemove = TRUE,
  convert = c(NA, "sparseChar"),
  remove = NULL,
  euroStyle = TRUE,
  sciIncl = TRUE,
  callFrom = NULL,
  silent = TRUE
)

Arguments

x vector to be converted
spaceRemove (logical) to remove all heading and trailing (white) space (until first non-space character)
convert (character) define which type of non-conform entries to convert to NAs. Note, if remove is selected to eliminate character-entries they cannot be converted any more. Use 'allChar' for all character-entries; 'sparseChar' sparse (ie rare) character entries; NA for converting 'Na' or 'na' to NA; if 'none' or NULL no conversions at all.
remove (character) define which type of non-conform entries to remove, removed items cannot converted to NA any more. Use 'allChar' for removing all character entries; NA for removing all instances of NA (except thiese created by converting text); all elements will be kept if 'none' or NULL.
correctToUnique

euroStyle (logical) if TRUE will convert all ’,’ (eg used as European decimal-separator) to ’.’ (as internally used by R as decimal-separator), thus allowing converting the European decimal format.

sciIncl (logical) include recognizing scientific notation (eg 2e-4)
callFrom (character) allow easier tracking of message(s) produced
silent (logical) suppress messages

Value
numeric vector (or matrix (if ’x’ is matrix))

See Also
numeric

Examples

x1 <- c("+4"," + 5","6","bb","Na","-7")
convToNum(x1,convert=c("allChar"))
convToNum(x1) # too many non-numeric instances for 'sparseChar'

x2 <- c("+4"," + 5","6","-7"," - 8","1e6","+ 2.3e4","-3E4","- 4E5")
convToNum(x2,convert=NA,remove=c("allChar",NA))
convToNum(x2,convert=NA,remove=c("allChar",NA),sciIncl=FALSE)

correctToUnique  Correct vector to unique

Description
correctToUnique checks ’x’ for unique entries, while maintaining the original length. If necessary a counter will added to non-unique entries.

Usage
correctToUnique(
  x,
  sep = " ",
  atEnd = TRUE,
  maxIter = 4,
  NAnum = TRUE,
  callFrom = NULL
)
countCloseToLimits

Arguments

- **x**: input character vector
- **sep**: (character) separator used when adding counter
- **atEnd**: (logical) decide location of placing the counter (at end or at beginning of initial text)
- **maxIter**: (numeric) max number of iterations
- **NAenum**: (logical) if TRUE NAs will be enumerated (NA_1, NA_2,...)
- **callFrom**: (character) for better tracking of use of functions

Value

character vector

See Also

unique will simply remove repeated elements, ie length of 'x' won’t remain constant, filtSizeUniq is more complex and slower, treatTxtDuplicates

Examples

correctToUnique(c("li0", "n", NA, NA, rep(c("li2", "li3"), 2), rep("n", 4)))

countCloseToLimits  Count from two vectors number of values close within given limits

Description

This functions summarizes the search of similar (or identical) numeric values from 2 initial vectors, it evaluates the result from initial search run by findCloseMatch(), whose output is a less convenient list. countCloseToLimits checks furthermore how many results within additional (more stringent) distance-limits may be found and returns the number of distance values within the limits tested. Designed for checking if threshold used with findCloseMatch() may be set more stringent, eg when searching reasonable FDR limits ...

Usage

countCloseToLimits(closeMatch, limitIdent = 5, prefix = "lim_")

Arguments

- **closeMatch**: (list) output from findCloseMatch(), ie list indicating which instances of 2 series of data have close matches
- **limitIdent**: (numeric) max limit or panel of threshold values to test (if single value, in addition a panel with values below will be tested)
- **prefix**: (character) prefix for names of output
Value

integer vector with counts for number of list-elements with at least one absolute value below threshold, names

See Also

findCloseMatch

Examples

set.seed(2019); aa <- sample(12:15,20,repl=TRUE) +round(runif(20),2)-0.5
bb <- 11:18
match1 <- findCloseMatch(aa,bb,com="diff",lim=0.65)
head(match1)
(tmp3 <- countCloseToLimits(match1,lim=c(0.5,0.35,0.2)))
(tmp4 <- countCloseToLimits(match1,lim=0.7))

countSameStartEnd

Count same start- and end- sites of edges (or fragments)

Description

Count same start- and end- sites of edges (or fragments)

The function is used to characterize partially redundant edges and accumulation of cutting/breakage sites.

Usage

countSameStartEnd(frag, minFreq = 2, nDig = 4)

Arguments

frag (matrix) 1st column beg start-sites, 2nd column end end-sites of edges, row-names to precise fragment identities are recommended

minFreq (integer) min number of accumulated sites for taking into account (allows filtering with large datasets)

nDig (integer) rounding: number of digits for columns beg.rat and end.rat in output

Value

matrix of 6 columns: input (beg and end), beg.n, beg.rat, end.n, end.rat

See Also

to build initial tree buildTree, contribToContigPerFrag, simpleFragFig
cutArrayInCluLike

Examples

frag1 <- cbind(beg=c(2,3,7,13,13,15,7,9,7, 3,3,5), end=c(6,12,8,18,20,20,19,12,12, 4,5,7))
rownames(frag1) <- letters[1:nrow(frag1)]
countSameStartEnd(frag1)
simpleFragFig(frag1)

cutArrayInCluLike(3-dim array in list of matrixes (or arrays) similar to organizing
into clusters)

Description

cutArrayInCluLike cuts 'dat' (matrix, data.frame or 3-dim array) in list (of appended lines) ac-
cording to 'cluOrg', which serves as instruction which line of 'dat' should be placed in which
list-element (like sorting according to cluster-numbers).

Usage

cutArrayInCluLike(dat, cluOrg, callFrom = NULL)

Arguments

dat array (3 dim)
cluOrg (factor) organization of lines to clusters
callFrom (character) allows easier tracking of message(s) produced

Value

list of matrixes (or arrays)

Examples

mat1 <- matrix(1:30,nc=3,dinames=list(letters[1:10],1:3))
cutArrayInCluLike(mat1,cluOrg=factor(c(2,rep(1:4,2),5)))
cutAtMultSites  
*Cut character-vector at multiple sites*

**Description**

This function cuts character vector after 'cutAt' (ie keep the search substring 'cutAt', different to `strsplit`). Used for theoretical enzymatic digestion (eg in proteomics)

**Usage**

```r
cutAtMultSites(y, cutAt)
```

**Arguments**

- `y`: character vector (better if of length=1, otherwise one won’t know which fragment stems from which input)
- `cutAt`: (character) search substring, ie 'cutting rule'

**Value**

modified (ie cut) character vector

**See Also**

`strsplit`, `nFragments0`, `nFragments`

**Examples**

```r
tmp <- "MSVSTMEDSCELVYTERIIAVSFPSTAEENFRSNLREVAQMLKSKHGNYLLFSERRPDITKLHAKVLEFGPDHLHTELEK"
cutAtMultSites(c(tmp,"ojioRij"),c("R","K"))
```

cutToNgrp  
*Cut numeric vector to n groups (ie convert to factor)*

**Description**

cutToNgrp is a more elaborate version of `cut` for cutting a the content of a numeric vector 'x' into a given number of groups, taken from the length of 'lev'. Besides, this function provides the group borders/limits for convention use with legends.

**Usage**

```r
cutToNgrp(x, lev, NAuse = FALSE, callFrom = NULL)
```
Arguments

- **x**: numeric vector
- **lev**: (character or numeric), the length of this argument tells the number of groups to be used for cutting
- **NAuse**: (logical) include NAs as separate group
- **callFrom**: (character) for better tracking of use of functions

Value

list with $grouped telling which element of 'x' goes in which group and $legTxt with group-borders for convenient use with legends

See Also

cut

Examples

```r
set.seed(2019); dat <- runif(30) + (1:30)/2
cutToNggrp(dat,1:5)
plot(dat,col=(1:5)[as.numeric(cutToNggrp(dat,1:5)$grouped)])
```

diffCombin

*Compute matrix of differences for all pairwise combinations of numeric vector*

Description

diffCombin returns matrix of differences (eg resulting from substitution) for all pairwise combinations of numeric vector 'x'.

Usage

diffCombin(x, diagAsNA = FALSE, prefix = TRUE, silent = FALSE, callFrom = NULL)

Arguments

- **x**: numeric vector to compute differences for all combinations
- **diagAsNA**: (logical) return all self-self combinations as NA (otherwise 0)
- **prefix**: (logical) if TRUE, dimnames of output will specify orientation (prefix='from.' and 'to.')</n- **silent**: (logical) suppress messages
- **callFrom**: (character) allow easier tracking of message(s) produced
Value

numeric matrix of all pairwise differences

See Also

diff for simple differences

Examples

diffCombin(c(10,11.1,13.3,16.6))

diffPPM

Description

diff()-like function to return difference in ppm between subsequent values. Result is oriented, ie neg ppm value means decrease (from higher to lower value). Note that if the absolute difference remains the same the difference in ppm will not remain same. Any difference to NA is returned as NA, thus a single NA will result in two NAs in output (unless NA is 1st or last).

Usage

diffPPM(dat, toPrev = FALSE, silent = FALSE, callFrom = NULL)

Arguments

dat (numeric) vector for calculating difference to preceeding/following value in ppm
toPrev (logical) determine orienationsilent (logical) suppress messagescallFrom (character) allows easier tracking of message(s) produced

Value

list with close matches of 'x' to given 'y', the numeric value dependes on 'sortMatch' (if FALSE then always value of 'y' otherwise of longest of x&y)

See Also

checkSimValueInSer and (from this package) .compareByDiff, diff

Examples

aa <- c(1000.01,1000.02,1000.05,1000.08,1000.09,1000.08)
.compareByPPM(list(aa,aa),30,TRUE) # tabular 'long' version
diffPPM(aa)
### Description

`elimCloseCoord` reduces number of rows in 'dat' by eliminating lines where x & y coordinates (columns of matrix 'dat' defined by 'useCol') are identical (overlay points) or very close. The stringency for 'close' values may be fine-tuned using `nDig`, this function uses internally `firstOfRepeated`.

### Usage

```r
elimCloseCoord(
  dat,
  useCol = 1:2,
  elimIdentOnly = FALSE,
  refine = 2,
  nDig = 3,
  callFrom = NULL,
  silent = FALSE
)
```

### Arguments

- **dat**: matrix (or data.frame) with main numeric input
- **useCol**: (numeric) index for numeric columns of 'dat' to use/consider
- **elimIdentOnly**: (logical) if TRUE, eliminate real duplicated points only (ie identical values only)
- **refine**: (numeric) allows increasing stringency even further (higher 'refine' .. more lines considered equal)
- **nDig**: (integer) number of significant digits used for rounding, if two 'similar' values are identical after this rounding the second will be eliminated.
- **callFrom**: (character) allows easier tracking of message(s) produced
- **silent**: (logical) suppress messages

### Value

resultant matrix/data.frame

### See Also

`findCloseMatch`, `firstOfRepeated`

### Examples

```r
da1 <- matrix(c(rep(0:4,5),0.01,1.1,2.04,3.07,4.5),nc=2); da1[,1] <- da1[,1]*99; head(da1)
elimCloseCoord(da1)
```
equLenNumber

**Equal character-length number**

**Description**

equLenNumber convert numeric entry 'x' to text, with all elements getting the same number of characters (ie by adding preceding or tailing 0s, if needed). So far, the function cannot handle scientific annotations.

**Usage**

```r
equLenNumber(x, silent = FALSE, callFrom = NULL)
```

**Arguments**

- `x` (character) input vector
- `silent` (logical) suppress messages
- `callFrom` (character) allows easier tracking of message(s) produced

**Value**

character vector formatted as equal number of characters per value

**See Also**

`sprintf`

**Examples**

```r
equLenNumber(c(12, -3, 321))
equLenNumber(c(12, -3.3, 321))
```

---

exclExtrValues

**Exclude extreme values (based on distance to mean)**

**Description**

This function aims to identify extreme values (values most distant to mean, thus potential outliers), mark them as NA or directly exclude them (depending on 'showNAs'). Note that every set of non-identical values will have at least one most extreme value. Extreme values are part of many distributions, they are not necessarily true outliers.
Usage

'exclExtrValues(
  dat,
  result = "val",
  CVlim = NULL,
  maxExcl = 1,
  showNA = FALSE,
  goodValues = TRUE,
  silent = FALSE,
  callFrom = NULL
)

Arguments

dat numeric vector, main input
result (character) may be 'val' for returning data without extreme values or 'pos' for returning position/index of extreme values
CVlim (NULL or numeric) allows to retain extreme values only if a certain CV (for all 'dat') is exceeded (to avoid calling extreme values form homogenous data-sets)
maxExcl (integer) max number of elements to exclude
showNA (logical) will display extreme values as NA
goodValues (logical) allows to display rather the good values instead of the extreme values
silent (logical) suppress messages
callFrom (character) allow easier tracking of message(s) produced

Value

numeric vector wo extreme values or index-position of extreme values

See Also

'firstOfRepLines', 'get1stOfRepeatedByCol' for treatment of matrix

Examples

x <- c(rnorm(30),-6,20)
exclExtrValues(x)
exponNormalize

Normalize by adjusting exponent

Description

This function normalizes 'dat' by optimizing exponent function (ie dat ^exp) to fit best to 'ref' (default: average of each line of 'dat').

Usage

exponNormalize(
  dat,
  useExpon,
  dynExp = TRUE,
  nStep = 20,
  startExp = 1,
  simMeas = "cor",
  refDat = NULL,
  refGrp = NULL,
  refLines = NULL,
  rSquare = FALSE,
  silent = FALSE,
  callFrom = NULL
)

Arguments

dat matrix or data.frame of numeric data to be normalized
useExpon (numeric vector or matrix) exponent values to be tested
dynExp (logical) require 'useExpon' as 2 values (matrix), will gradually increase exponent from 1st to 2nd; may be matrix or data.frame for dynamic, in this case 1st line for exp for lowest data, 2nd line for highest
nStep (integer) number of exponent variations (steps) when testing range from-to
startExp (numeric)
simMeas (character) similarity metric to be used (so far only "cor"), if rSquare=TTRUE, the r-squared will be returned
refDat (matrix or data.frame) if null average of each line from 'dat' will be used as reference in similarity measure
refGrp (factor) designing which col of 'ref' should be used with which col of 'dat' (length equal to number of cols in 'dat'). Note: 'refGrp' not yet coded optimally to extract numeric part of character vector, protential problems when all lines or cols of dat are NA
refLines (NULL or integer) optional subset of lines to be considered (only) when determining normalization factors
extr1chan

+rSquare: (logical) if TRUE, add r-squared
+silent: (logical) suppress messages
+callFrom: (character) allow easier tracking of message(s) produced

**Value**

matrix of normalized data

**See Also**

more evolved than `normalizeThis` with argument set to 'exponent'

**Examples**

```r
set.seed(2016); dat1 <- matrix(c(runif(200)+rep(1:10,20)),nc=10)
head(rowGrpCV(dat1,gr=gl(4,3,labels=LETTERS[1:4])[2:11]))
set.seed(2016); dat1 <- c(0,1,0,2,0,3,0.5)*rep(c(1,10),each=4)
dat1 <- matrix(round(c(sqrt(dat1),dat1^1.5,3*dat1+runif(length(dat1))),2),nc=3)
dat2a <- exponNormalize(dat1[,1],useExpon=2,nSte=1,refD=dat1[,3])
layout(matrix(1:2,nc=2))
plot(dat1[,1],dat1[,3],type="b",main="init",ylab="ref")
plot(dat2a$datNor[,1],dat1[,3],type="b",main="norm",ylab="ref")
dat2b <- exponNormalize(dat1[,1],useExpon=c(1.7,2.3),nSte=5,refD=dat1[,3])
plot(dat1[,1],dat1[,3],type="b",main="init",ylab="ref")
plot(dat2b$datNor[,1],dat1[,3],type="b",main="norm",ylab="ref")
dat2c <- exponNormalize(dat1[-3],useExpon=matrix(c(1.7,2.3,0.6,0.8),nc=2),nSte=5,refD=dat1[,3]);
plot(dat1[,1],dat1[,3],type="b",main="init",ylab="ref")
plot(dat2c$datNor[,1],dat1[,3],type="b",main="norm 1",ylab="ref")
plot(dat1[,2],dat1[,3],type="b",main="init",ylab="ref")
plot(dat2c$datNor[,2],dat1[,3],type="b",main="norm 2",ylab="ref");
```

extract one series, ie channel, of list of arrays

**Description**

This function was designed for handling measurements stored as list of multiple arrays, like eg compound-screens using microtiter-plates where multiple parameters ('channels') were recorded for each well (element). The elements (eg compounds screened) are typically stored in the 1st dimension of the arrays, the replicated in the secon dimension and different measure types/parameters in the 3rd channel. In order to keep the structure of of individual microtiter-plates, typically each plate forms a separate array (of same dimensions) in a list. The this function allows extracting a single channel of the list of arrays (3rd dim of each array) and return row-appended matrix.

**Usage**

```r
extr1chan(arrLst, cha, na.rm = TRUE, rowSep = "__")
```
extractLast2numericParts

Extract last two numeric parts from character vector

Description

extractLast2numericParts extracts last 2 (integer) numeric parts between punctuations out of character vector 'x'. Runs faster than gregexpr. Note: won't work correctly with decimals or exponential signs !! (such characters will be considered as punctuation, ie as separator)

Usage

extractLast2numericParts(x, silent = FALSE, callFrom = NULL)

Arguments

x main character input
silent (logical) suppres messages
callFrom (character) allow easier tracking of message(s) produced

Value

(numeric) matrix with 2 columns (eg from initial concatenated coordinates)
extrColsDeX

See Also
gregexpr from grep

Examples

extractLast2numericParts(c("M01.1-4","M001/2.5","M_0001_03-16","zyx","012","a1.b2.3-7,2"))

extrColsDeX Flexible extraction of columns

Description

This function provides flexible checking if a set of columns may be extracted from a matrix or data.frame 'x'.

Usage

extrColsDeX(x, extrCol, doExtractCols = FALSE, callFrom = NULL, silent = FALSE)

Arguments

x (matrix or data.frame) min input
extrCol (character or integer) columns to be extracted, may be column-names or column index
doExtractCols (logical) if default FALSE only the column indexes will be returned
callFrom (character) allows easier tracking of message(s) produced
silent (logical) suppress messages

Value

integer-vector (if doExtractCols=FALSE return depending on input matrix or data.frame)

See Also

read.table, filterList

Examples

dFr <- data.frame(a=11:14,b=24:21,cc=LETTERS[1:4],dd=rep(c(TRUE,FALSE),2))
extrColsDeX(dFr,c("b","cc","notThere"))
extrColsDeX(dFr,c("b","cc","notThere"),doExtractCols=TRUE)
extrNumericFromMatr  Extract numeric part of matrix or data.frame

Description
extrNumericFromMatr extracts numeric part of matrix or data.frame, removing remaining non-numeric elements if trimToData is set to TRUE. Note, that cropping entire lines where a (single) text element appeared may quickly reduce the overall content of the input data.

Usage
extrNumericFromMatr(dat, trimToData = TRUE, silent = FALSE, callFrom = NULL)

Arguments
- dat: matrix (or data.frame) for extracting numeric parts
- trimToData: (logical) default to remove (crop) lines and cols contributing to NA, non-numeric data is transformed to NA
- silent: (logical) suppress messages
- callFrom: (character) allow easier tracking of message(s) produced

Value
matrix of numeric data

Examples
mat <- matrix(c(letters[1:7],14:16,LETTERS[1:6]),nrow=4,dimnames=list(1:4,letters[1:4]))
mat; extrNumericFromMatr(mat)

extrSpcText  Extract specific text

Description
extrSpcText extracts/cuts text-fragments out of 'txt' following specific anchors 'cutFrom' and 'cutTo'. In case 'cutFrom' not found 'missingAs' will be returned. In case 'cutTo' not found text gets extracted with 'chaMaxEl' characters.
filt3dimArr

Filter a three-dimensional array of numeric data

Description

Filtering of 3-dim array ('x') : filter column 'filtCrit' as 'larger as' (according to 'filtTy') 'filtVal' and extract/display all col matching 'displCrit'.

Usage

filt3dimArr(x, filtCrit, filtVal, filtTy = ">", displCrit = NULL)
Arguments

- **x**: array (3-dim) of numeric data
- **filtCrit**: (character, length=1) which column-name consider when filtering filter with 'filtVal' and 'filtTy'
- **filtVal**: (numeric) for testing inferior/superior/equal condition
- **filtTy**: (character) which type of testing to perform ('eq', 'inf', 'infeq', 'sup', 'supeq', '> ', '< ', '>= ', '< = ', '== ')
- **displCrit**: (character) column-name(s) to display

Value

list of filtered matrixes (by 3rd dim)

Examples

```r
arr1 <- array(1:24, dim=c(4,3,2), dimnames=list(c(LETTERS[1:4]), paste("col",1:3,sep=""),c("ch1","ch2")))
filt3dimArr(arr1, displCrit=c("col1","col2"), filtCrit="col2", filtVal=7)
```

---

**filterList**

*Filter for unique elements*

Description

This function aims to identify and remove duplicated elements in a list and maintain the list-structure in the output. `filtSizeUniq` filters 'lst' (list of character-vectors or character-vector) for elements being unique (to 'ref' or if NULL to all 'lst') and of character length. In addition, the min- and max- character length may be filtered, too. Eg, in proteomics this helps removing peptide sequences which would not be measured/detected any way.

Usage

```r
filterList(lst, filt, minLineRatio = 0.5, silent = FALSE, callFrom = NULL)
```

Arguments

- **lst**: (list) main input, each vector, matrix or data.frame in this list will be filtered if its length or number of lines fits to filt
- **filt**: (logical) vector of FALSE/TRUE to use for filtering. If this a matrix is given, the value of minLineRatio will be applied as threshold of min content of TRUE for each line of filt
- **minLineRatio**: (numeric) in case filt is a matrix of FALSE/TRUE, this value will be used as threshold of min content of TRUE for each line of filt
- **silent**: (logical) suppress messages
- **callFrom**: (character) allow easier tracking of message(s) produced
filtSizeUniq

Filter for unique elements

Value

list of filtered input

See Also

correctToUnique, unique, duplicated, extrColsDeX

Examples

set.seed(2020); dat1 <- round(runif(80),2)
list1 <- list(m1=matrix(dat1[1:40],ncol=8),m2=matrix(dat1[41:80],ncol=8),other=letters[1:8])
rownames(list1$m1) <- rownames(list1$m2) <- paste0("line",1:5)
filterList(list1, list1$m1[,1] >0.4)
filterList(list1, list1$m1 >0.4)

filtSizeUniq

Description

This function aims to identify and remove duplicated elements in a list and maintain the list-
structure in the output. filtSizeUniq filters 'lst' (list of character-vectors or character-vector)
for elements being unique (to 'ref' or if NULL to all 'lst') and of character length. In addition, the
min- and max- character length may be filtered, too. Eg, in proteomics this helps removing peptide
sequences which would not be measured/detected any way.

Usage

filtSizeUniq(
  lst,
  ref = NULL,
  minSize = 6,
  maxSize = 36,
  filtUnique = TRUE,
  byProt = TRUE,
  inclEmpty = TRUE,
  silent = FALSE,
  callFrom = NULL
)

Arguments

lst list of character-vectors or character-vector

ref (character) optional alternative 'reference', if not NULL used in addition to 'lst'

for considering elements of 'lst' as unique

minSize (integer) minimum number of characters, if NULL set to 0

maxSize (integer) maximum number of characters
filtUnique  (logical) if TRUE return unique-only character-strings
byProt    (logical) if TRUE organize output as list (by names of input, eg protein-names) -
           if 'lst' was named list
inclEmpty (logical) optional including empty list-elements when all elements have been
           filtered away - if 'lst' was named list
silent    (logical) suppress messages
callFrom  (character) allow easier tracking of message(s) produced

Value
list of filtered input

See Also
    correctToUnique, unique, duplicated

Examples
    filtSizeUniq(list(A="a",B=c("b","bb","c"),D=c("dd","d","ddd","c")),filtUn=TRUE,minSi=NULL)
    # input: c and dd are repeated
    filtSizeUniq(list(A="a",B=c("b","bb","c"),D=c("dd","d","ddd","c")),ref=c(letters[1:26,1:3],
    "dd","dd","bb","ddd"),filtUn=TRUE,minSi=NULL) # a,b,c,dd repeated

findCloseMatch  Find close numeric values between two vectors

Description
findCloseMatch finds close matches (similar values) between two numeric vectors ('x','y') based
on method 'compTy' and threshold 'limit'. Return list with close matches of 'x' to given 'y', the
numeric value dependes on 'sortMatch' (if FALSE then always value of 'y' otherwise of longest of
x&y). Note: Speed & memory improvement if 'sortMatch'=TRUE (but result might be inversed!):
adopt search of x->y or y->x to searching matches of each longest to each shorter (ie flip x &y).
Otherwise, if length of 'x' & 'y' are very different, it may be advantageous to use a long(er) 'x' and
short(er) 'y' (with 'sortMatch'=FALSE). Note: Names of 'x' & 'y' or (if no names) prefix letters
'x' & 'y' are always added as names to results.

Usage
findCloseMatch(
    x,
    y,
    compTy = "ppm",
    limit = 5,
    asIndex = FALSE,
    maxFitShort = 100,
    sortMatch = FALSE,
findCloseMatch

silent = FALSE,
callFrom = NULL
)

Arguments

x numeric vector for comparison
y numeric vector for comparison
compTy (character) may be 'diff' or 'ppm', will be used with threshold from argument 'limit'
limit (numeric) threshold value for retaining values, used with distace-type specified in argument 'compTy'
asIndex (logical) optionally rather report index of retained values
maxFitShort (numeric) limit output to max number of elements (avoid returning high number of results if filtering was not enough stringent)
sortMatch (logical) if TRUE than matching will be preformed as 'match longer (of x & y) to closer', this may process slightly faster (eg 'x' longer: list for each 'y' all 'x' that are close, otherwise list of each 'x'),
silent (logical) suppress messages
callFrom (character) allow easier tracking of message(s) produced

Value

list with close matches of 'x' to given 'y', the numeric value dependes on 'sortMatch' (if FASLE then always value of 'y' otherwise of longest of x&y)

See Also

checkSimValueInSer and (from this package) .compareByDiff, for convient output countCloseToLimits

Examples

aa <- 11:14 ; bb <- c(13,1,11.5,14,3,20,21)
findCloseMatch(aa,bb,com="diff",lim=0.6)
findCloseMatch(c(a=5,b=11,c=12,d=18),c(G=2,H=11,I=12,J=13)+0.5,comp="diff",lim=2)
findCloseMatch(c(4,5,11,12,18),c(2,11,12,13,33)+0.5,comp="diff",lim=2)
findCloseMatch(c(4,5,11,12,18),c(2,11,12,13,33)+0.5,comp="diff",lim=2,sort=FALSE)
.compareByDiff(list(c(a=10,b=11,c=12,d=13),c(H=11,I=12,J=13,K=33)+0.5),limit=1)  # return matrix

a2 <- c(11:20); names(a2) <- letters[11:20]
b2 <- c(25:5)+c(rep(0,5),(1:10)/50000,rep(0,6)); names(b2) <- LETTERS[25:5]
which(abs(b2-a2[8]) < a2[8]*1e-6) # find R=18 : no10
findCloseMatch(a2,b2,com="ppm",lim=5)  # find Q,R,S,T
findCloseMatch(a2,b2,com="ppm",lim=5,asI=TRUE)  # find Q,R,S,T
findCloseMatch(b2,a2,com="ppm",lim=5,asI=TRUE,sort=FALSE)  # find Q,R,S,T
findCloseMatch(a2,b2,com="ratio",lim=1.000005)  # find Q,R,S,T
findCloseMatch(a2,b2,com="diff", lim=0.00005)  # find S,T
findRepeated  

*Find repeated elements*

**Description**

findRepeated gets index of repeated items/values in vector 'x' (will be treated as character). Return (named) list of indexes for each of the repeated values, or NULL if all values are unique. This approach is similar but more basic compared to get1stOfRepeatedByCol.

**Usage**

```
findRepeated(x, nonRepeated = FALSE, silent = FALSE, callFrom = NULL)
```

**Arguments**

- `x` character vector
- `nonRepeated` (logical) if =TRUE, return list with elements $rep$ and $nonrep$
- `silent` (logical) suppress messages
- `callFrom` (character) allow easier tracking of message(s) produced

**Value**

(named) list of indexes for each of the repeated values, or NULL if all values unique

**See Also**

similar approach but more basic than get1stOfRepeatedByCol

**Examples**

```
aa <- c(11:16,14:12,14); findRepeated(aa)
```

findSimilFrom2sets  

*Find similar numeric values from two vectors/matrixes*

**Description**

findSimilFrom2sets compares to vectors or matrixes and returns combined view including only all close (by findCloseMatch). Return matrix (predMatr) with add'l columns for index to and 'grp' (group of similar values (1-to-many)), 'nGrp' (n of grp), 'isBest' or 'nBest', 'disToMeas' (distance/difference between pair) & 'ppmToPred' (distance in ppm). Note: too wide 'limitComp' will result in large window and many 'good' hits will compete (and be mutually excluded) if selection 'bestOnly' is selected.
Usage

```r
findSimilFrom2sets(
  predMatr,
  measMatr,
  colMeas = 1,
  colPre = 1,
  compareTy = "diff",
  limitComp = 0.5,
  bestOnly = FALSE,
  silent = FALSE,
  callFrom = NULL,
  debug = FALSE
)
```

Arguments

- **predMatr** (matrix or numeric vector) dataset number 1, referred to as 'predicted', the column specified in argument `colPre` points to the data to be used.
- **measMatr** (matrix or numeric vector) dataset number 2, referred to as 'measured', the column specified in argument `colMeas` points to the data to be used.
- **colMeas** (integer) which column number of 'measMatr' to consider.
- **colPre** (integer) which column number of 'predMatr' to consider.
- **compareTy** (character) 'diff' (difference) 'ppm' (relative difference).
- **limitComp** (numeric) limit used by 'compareTy'.
- **bestOnly** (logical) allows to filter only hits with min distance (defined by 'compareTy'). 3rd last col will be 'nBest' - otherwise 3rd last col 'isBest'.
- **silent** (logical) suppress messages.
- **callFrom** (character) allow easier tracking of message(s) produced.
- **debug** (logical) for bug-tracking: more/enhanced messages.

Value

matrix (predMatr) with add'l columns for index to and 'grp' (group of similar values (1-to-many)), 'nGrp' (n of grp), 'isBest' or 'nBest', 'disToMeas' (distance/difference between pair) & 'ppmToPred' (distance in ppm).

See Also

- `checkSimValueInSer`
- `findCloseMatch`
- `closeMatchMatrix`

Examples

```r
aA <- c(11:17); bB <- c(12.001,13.999); cC <- c(16.2,8,9,12.5,12.6,15.9,14.1)
aZ <- matrix(c(aA,aA+20),ncol=2,dimnames=list(letters[1:length(aA)],c("aaA","aZ")))
cZ <- matrix(c(cC,cC+20),ncol=2,dimnames=list(letters[1:length(cC)],c("ccC","cZ")))
findCloseMatch(cC,aA,com="diff",lim=0.5,sor=FALSE)
```
findUsableGroupRange

Select groups within given range

Description

This function aims to help finding stretches/segments of data with a given maximum number of NA-instances. This function is used to inspect/filter each lines of 'dat' for a subset with sufficient presence/absence of NA values (ie limit number of NAs per level of 'grp'). Note: optimal performance with n.lines » n.groups

Usage

findUsableGroupRange(dat, grp, maxNA = 1, callFrom = NULL)

Arguments

dat (matrix or data.frame) main input

grp (factor) information which column of 'dat' is replicate of whom

maxNA (integer) max number of tolerated NAs

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

Value

matrix with boundaries of 1st and last usable column (NA if there were no suitable groups found)

Examples

dat1 <- matrix(1:56,nc=7)
dat1[c(2,3,4,5,6,10,12,18,19,20,22,23,26,27,28,30,31,34,38,39,50,54)] <- NA
rownames(dat1) <- letters[1:nrow(dat1)]
findUsableGroupRange(dat1,gl(3,3)[-3:4])
firstLineOfDat  Filter matrix to keep only first of repeated lines

Description

This function aims to reduce the complexity of a matrix (or data.frame) in case column 'refCol' has multiple lines with same value. In this case, it reduces the input-data to 1st line of redundant entries and returns a matrix (or data.frame) without lines identified as redundant entries for 'refCol'). in sum, this functions works like using unique on a given column, and propagates the same treatment to all other columns.

Usage

firstLineOfDat(dat, refCol = 2, silent = FALSE, callFrom = NULL)

Arguments

dat (matrix or data.frame) main input
refCol (integer) column number of reference-column
silent (logical) suppress messages
callFrom (character) allow easier tracking of message(s) produced

Value

matrix (same number of columns as input)

See Also

cfirstOfRepeated, unique, duplicated

Examples

(mat1 <- matrix(c(1:6,rep(1:3,1:3)),ncol=2,dimnames=list(letters[1:6],LETTERS[1:2])))
firstLineOfDat(mat1)

Description

This function works similar to unique, but provides additional information about which elements of original input 'x' are repeated by providing indexes relative to the input. firstOfRepeated makes list with 3 elements : $indRepeated.. index for first of repeated 'x', $indUniq.. index of all unique + first of repeated, $indRedund.. index of all redundant entries, ie non-unique (wo 1st). Used for reducing data to non-redundant status, however, for large numeric input the function nonAmbiguousNum() may perform better/faster. NAs won't be considered (NAs do not appear in reported index of results), see also firstOfRepLines().
Usage

firstOfRepeated(x)

Arguments

x (character or numeric) main input

Value

list with indices: $\text{indRepeated}, \text{indUniq}, \text{indRedund}$

See Also

duplicated, nonAmbiguousNum, firstOfRepLines gives less detail in output (lines/elements/indexes of omitted not directly accessible) and works faster

Examples

```r
x <- c(letters[c(3,2:4,8,NA,3:1,NA,5:4)]); names(x) <- 100+(1:length(x))
firstOfRepeated(x)
x[firstOfRepeated(x)$indUniq]  # only unique with names
```

---

firstOfRepLines Reduce to first occurrence of repeated lines

Description

This function concatenates all columns of input-matrix and then searches like unique for unique elements, optionally the indexes of unique elements may get returned. Note: This function reads input as character (thus won’t understand 10==10.0). Returns simplified/non-redundant vector/matrix (i.e. fewer lines), or respective index. faster than firstOfRepeated

Usage

```r
firstOfRepLines(mat, outTy = "ind", useCol = NULL, callFrom = NULL)
```

Arguments

mat initial matrix to treat
outTy for output type: ‘ind’.. index to 1st occurrence (non-red),’orig’..non-red lines of mat, ‘conc’.. non-red concatenated values, ‘num’.. index to which group/category the lines belong
useCol (integer) custom choice of which columns to paste/concatenate
callFrom (character) allows easier tracking of messages produced

Value

simplified/non-redundant vector/matrix (i.e fewer lines for matrix), or respective index
fuseAnnotMatr

See Also

unique, nonAmbiguousNum, faster than firstOfRepeated which gives more detail in output (lines/elements/indexes of omitted)

Examples

```r
mat <- matrix(c("e", "n", "a", "n", "z", "z", "n", "z", "z", "b", "n", "c", "n", "n", "n", "n", "n", "z"), ncol=2)
firstOfReplLines(mat, out="conc")
```

Description

In a number of instances experimental measurements and additional information (annotation) are provided by separate objects (matrices) as they may not be generated the same time. The aim of this function is provide help when matching appropriate lines for 2 sets of data (experimental measures in iniTab and annotation from annotTab) for fusing. fuseAnnotMatr adds suppelmental columns/annotation to an initial matrix iniTab : using column 'refIniT' as key (in iniTab) to compare with key 'refAnnotT' (from 'annotTab'). The columns to be added from annotTab must be chosen explicitly. Note: if non-unique IDs in iniTab : runs slow (but save) due to use of loop for each unique ID.

Usage

```r
fuseAnnotMatr(
  iniTab, annotTab,
  refIniT = "Uniprot", refAnnotT = "combName",
  addCol = c("ensembl_gene_id", "description", "geneName", "combName"),
  debug = TRUE, silent = FALSE, callFrom = NULL
)
```

Arguments

- `iniTab` (matrix), that may have lines with multiple (=repeated) key entries
- `annotTab` (matrix) containing reference annotation
- `refIniT` (character) type of reference (eg 'Uniprot')
- `refAnnotT` (character) column name to use for reference-annotation
- `addCol` (character) column-names of 'annotTab' to use/extract (if no matches found, use all)
- `debug` (logical) for bug-tracking: more/enhanced messages
fuseCommonListElem

Fuse content of list-elements with redundant (duplicated) names

Description

fuseCommonListElem fuses (character or numeric) elements of list re-occuring under same name, so that resultant list has unique names. Note: will not work with list of matrixes

Usage

fuseCommonListElem(
  lst,
  initOrd = TRUE,
  removeDuplicates = FALSE,
  callFrom = NULL
)

Arguments

lst (list) main input, list of numeric vectors
initOrd (logical) preserve initial order in output (if TRUE) or otherwise sort alphabetically
removeDuplicates (logical) allow to remove duplicate entries (if vector contains names, both the name and the value need to be identical to be removed; note: all names must have names with more than 0 characters to be considered as names)
callFrom (character) allows easier tracking of message(s) produced
fusePairs

Value

fused list (same names as elements of input)

See Also

unlist

Examples

val1 <- 10 + 1:26
names(val1) <- letters
lst1 <- list(c=val1[3:6],a=val1[1:3],b=val1[2:3],a=val1[12],c=val1[13])
fuseCommonListElem(lst1)

---

fusePairs

Fuse pairs to generate cluster-names

Description

Fuse previously identified pairs to ’clusters’, return vector with cluster-numbers.

Usage

fusePairs(
  datPair,
  refDatNames = NULL,
  inclRepLst = FALSE,
  maxFuse = NULL,
  debug = FALSE,
  silent = TRUE,
  callFrom = NULL
)

Arguments

datPair  2-column matrix where each line represents 1 pair
refDatNames (NULL or character) allows placing selected pairs in context of larger data-set (names to match those of ’datPair’)
inclRepLst  (logical) if TRUE, return list with ’clu’ (clu-numbers, default output) and ’refLst’ (list of clustered elements, only n>1)
maxFuse  (integer, default NULL) maximal number of groups/clusters
debug  (logical) for bug-tracking: more/enhanced messages and intermediate objects written in global name-space
silent  (logical) suppress messages
callFrom  (character) allow easier tracking of message(s) produced
get1stOfRepeatedByCol

Description

get1stOfRepeatedByCol sorts matrix 'mat' and extracts only 1st occurance of values in column 'sortBy'. Returns then non-redundant matrix (ie for column 'sortBy', if 'markIfAmbig' specifies existing col, mark ambig there). Note : problem when sortSupl or sortBy not present (or not intended for use)

Usage

get1stOfRepeatedByCol(
  mat,
  sortBy = "seq",
  sortSupl = "ty",
  asFirstLast = c("full", "inter"),
  markIfAmbig = c("ambig", "seqNa"),
  asList = FALSE,
  abmiPref = "_"
)

Arguments

  mat (matrix or data.frame) numeric vector to be tested
  sortBy column name for which elements should be made unique, numeric or character column; 'sortSupl' .. add'l colname to always select specific 1st)
  sortSupl default="ty"
  asFirstLast (character,length=2) to force specific strings from colunm 'sortSupl' as first and last when selecting 1st of repeated terms, default=c("full","inter")
  markIfAmbig (character,length=2) 1st will be set to 'TRUE' if ambiguous/repeated, 2nd will get (heading) prefix, default=c("ambig","seqNa")
  asList (logical) to return list with non-redundant ('unique') and removed lines ('repeats')
  abmiPref (character) prefix to note ambiguous entries/terms, default="_"
**getValuesByUnique**

---

**Value**

depending on 'asList' either list with non-redundant ('unique') and removed lines ('repeats')

**See Also**

firstOfRepeated for (more basic) treatment of simple vector, nonAmbiguousNum for numeric use (much faster !!!)

**Examples**

```r
aa <- cbind(no=as.character(1:20), seq=sample(LETTERS[1:15],20, repl=TRUE), 
ty=sample(c("full","Nter","inter"),20, repl=TRUE), ambig=rep(NA,20), seqNa=1:20)
get1stOfRepeatedByCol(aa)
```

---

**Description**

When data have repeated elements (defined by names inside the vector), it may be advantageous to run some operations only on a unique set of the initial data, or sometimes all repeated occurrences need to be replaced by a common (summarizing) value. This function allows to re-introduce new values from a second vector with unique names, to return a final vector of initial input-length and order of names (elements) like initial, too. Normally the user would provide 'datUniq' (without repeated names) containing new values which will be expanded to structure of 'dat', if 'datUniq' is not provided a vector with unique names will be made using the first occurrence of repeated value(s). For more complex cases the indexing relative to 'datUniq' can be returned (setting asIndex=TRUE). Note: If not all names of 'dat' are found in 'datUniq' the missing spots will be returned as NA.

**Usage**

```r
getValuesByUnique(
  dat, 
  datUniq = NULL, 
  asIndex = FALSE, 
  silent = FALSE, 
  callFrom = NULL
)
```

**Arguments**

- `dat` (numeric or character) main long input, must have names
- `datUniq` (numeric or character) will be used to impose values on dat, must have names that should match names (at least partially) from dat
- `asIndex` (logical) if TRUE index values will be returned instead of replacing values
- `silent` (logical) suppress messages
- `callFrom` (character) allow easier tracking of message(s) produced
Value

vector of length dat with imposed values, or index values if asIndex=TRUE

See Also

unique, findRepeated, correctToUnique, treatTxtDuplicates

Examples

dat <- 11:19
names(dat) <- letters[c(6:3,2:4,8,3)]
## let's make a 'datUniq' with the mean of repeated values :
datUniq <- round(tapply(dat,names(dat),mean),1)
## now propagate the mean values to the full vector
getValuesByUnique(dat,datUniq)
getValuesByUnique(dat,datUniq)

-------------------

htmlSpecCharConv

**Html special character conversion**

Description

Converts 'txt' so that special characters (like 'beta','micro','square' etc) will be displayed correctly when used for display in html (eg at mouse-over). Note : uses package stringi

Usage

htmlSpecCharConv(txt)

Arguments

txt character vector including special characters

Value

corrected character vector adopted to html display

See Also


Examples

(x <- stringi::stri_unescape_unicode("\u00b5\u003d\u0061\u0062"))
htmlSpecCharConv(x)
Fit linear regression, return parameters and p-values

Description

This function fits a linear regression and returns the parameters, including p-values from Anova. Here the vector 'y' (scalar response or dependent variable, i.e., the value that should get estimated) will be estimated according to 'dep' (explanatory or independent variable). Alternatively, 'dep' may me a matrix where 1st column will be used as 'dep and the 2nd column as 'y'.

Usage

linRegrParamAndPVal(dep, y = NULL, asVect = TRUE)

Arguments

dep (numeric vector, matrix or data.frame) explanatory or dependent variable, if matrix or data.frame the 1st column will be used, if 'y'=NULL the 2nd column will be used as 'y'

y (numeric vector) independent variable (the value that should get estimated based on 'dep')

asVect (logical) return numeric vector (Intercept, slope, p.intercept, p.slope) or matrix or results

Value

numeric vector (Intercept, slope, p.intercept, p.slope), or if asVect==TRUE as matrix (p.values in 2nd column)

See Also

lm

Examples

linRegrParamAndPVal(c(5,5.1,8,8.2),gl(2,2))
listBatchReplace  

Replacements in list

Description

listBatchReplace replaces in list 'lst' all entries with value 'searchValue' by 'replaceBy'

Usage

listBatchReplace(lst, searchValue, replaceBy, silent = FALSE, callFrom = NULL)

Arguments

- **lst**: input-list to be used for replacing
- **searchValue**: (character, length=1)
- **replaceBy**: (character, length=1)
- **silent**: (logical) suppress messages
- **callFrom**: (character) allow easier tracking of message(s) produced

Value

corrected list

See Also

basic replacement sub in grep

Examples

```r
lst1 <- list(aa=1:4,bb=c("abc","efg","abhh","effge"),cc=c("abdc","efg"))
listBatchReplace(lst1,search="efg",repl="EFG",sil=FALSE)
```

listGroupsByNames  

Organize values into list and sort by names

Description

Sort values of 'x' by its names and organize as list by common names, the names until 'sep' are used for (re)grouping. Note that typical spearators occuring the initial names may need protection by '\' (this is automatically taken care of for the case of the dot ('.') separator).

Usage

```r
listGroupsByNames(x, sep = ".", silent = FALSE, callFrom = NULL)
```
Arguments

- **x**  
  (list) main input
- **sep**  
  (character) separator (note that typical separators may need to be protected, only automatically added for '.')
- **silent**  
  (logical) suppress messages
- **callFrom**  
  (character) allows easier tracking of message(s) produced

Value

matrix or data.frame

See Also

`rbind` in `cbind`

Examples

```r
listGroupsByNames((1:10)/5)
s1 <- 1:6; names(s1) <- c("AA","BB","AA.1","CC","AA.b","BB.e")
listGroupsByNames(s1)
```

---

**lmSelClu**  
Run lm on segmented data (from clustering)

Description

`lmSelClu` runs linear regression on data segmented previously (eg by clustering). This function offers various types of (2-coefficient) linear regression on 2 columns of 'dat' (matrix with 3rd col named 'clu' or 'cluID', numeric elements for cluster-number). If argument 'clu' is (default) 'max', the column 'clu' will be inspected to take most frequent value of 'clu', otherwise a numeric entry specifying the cluster to extract is expected. Note: this function was initially made for use with results from `diagCheck()` Note: this function lacks means of judging godness of fit of the regression performed & means for plotting

Usage

```r
lmSelClu(
  dat,
  useCol = 1:2,
  clu = "max",
  regTy = "lin",
  filt1 = NULL,
  filt2 = NULL,
  silent = FALSE,
  callFrom = NULL
)
```
**Arguments**

- **dat**: matrix or data.frame
- **useCol**: (integer or character) specify which 2 columns of `dat` to use for linear regression
- **clu**: (character) name of cluster to be extracted and treated
- **regTy**: (character) change type used for linear regression: 'lin' for 1st col ~ 2nd col, 'res' for residue ~ 2nd col, 'norRes' for residue/2nd col ~ 2nd col or 'sqNorRes', 'inv' for 1st col ~ 1/(2nd col), 'invRes' for residue ~ 1/(2nd col)
- **filt1**: (logical or numerical) filter criteria for 1st of `useCol`, if numeric then select all lines of dat less than max of filt1
- **filt2**: (logical or numerical) filter criteria for 2nd of `useCol`, if numeric then select all lines of dat less than max of filt2
- **silent**: (logical) suppress messages
- **callFrom**: (character) allows easier tracking of message(s) produced

**Value**

lm object (or NULL if no data left)

**See Also**

lm

**Examples**

```r
set.seed(2016); ran1 <- runif(220)
mat1 <- round(rbind(matrix(c(1:100+ran1[1:100],rep(1,50)),ncol=3),
    matrix(c(1:60,68:9+ran1[101:160],rep(2,60)),nc=3)),1)
colnames(mat1) <- c("a","BB","clu")
lmSelClu(mat1)
plot(mat1[which(mat1[,3]=="2"),1:2],col=grey(0.6))
abline(lmSelClu(mat1),lty=2,lwd=2)
#
mat2 <- round(rbind(matrix(c(1:100+ran1[1:100],rep(1,50)),ncol=3),
    matrix(c(1:60,(2:61+ran1[101:160])^2,rep(2,60)),nc=3)),1)
colnames(mat2) <- c("a","BB","clu")
(reg2 <- lmSelClu(mat2,regTy="sqNor"))
plot(function(x) coef(reg2)[2]+ (coef(reg2)[2]*x^2),xlim=c(1,70))
points(mat2[which(mat2[,3]=="2"),1:2],col=2)
```

**Description**

`lrbind` is `rbind` on lists.

`lrbind`-like function to append list-elements containing tables and return one long table. Accepts also list-entries with data.frames or vectors (of length no of columns) as long as at least 1 list-entry is a matrix.
**Usage**

```r
lrbind(lst, silent = FALSE, callFrom = NULL)
```

**Arguments**

- `lst` (list): main input (each list-element should have same number of columns, numeric vectors will be converted to number of columns of other elements)
- `silent` (logical): suppress messages
- `callFrom` (character): allow easier tracking of message(s) produced

**Value**

matrix or data.frame

**See Also**

`rbind` in `cbind`

**Examples**

```r
lst1 <- list(matrix(1:9,nc=3,dimnames=list(letters[1:3],c("AA","BB","CC"))),
             1:13,matrix(51:56,ncol=3))
lrbind(lst1)
```

---

**Description**

`makeMAList` extracts sets of data-pairs (like R & G series) and makes MA objects as MA-List object (eg for ratio oriented analysis). The grouping of columns as sets of replicate-measurements is done according to argument 'MAfac'. The output is fully compatible to functions of package `limma` (Bioconductor).

**Usage**

```r
makeMAList(
    mat,
    MAfac,
    useF = c("R", "G"),
    isLog = TRUE,
    silent = FALSE,
    callFrom = NULL
  )
```
makeNRedMatr

Make non-redundant matrix

Description

makeNRedMatr takes matrix or data.frame 'dat' to summarize redundant lines (column argument iniID) along method specified in summarizeRedAs to treat all lines with redundant iniID by same approach (ie for all columns the line where specified column is at eg max = 'maxOfRef' ). If no name given, the function will take the last numeric (factors may be used - they will be read as levels).

Usage

makeNRedMatr(
  dat,
  summarizeRedAs,
  iniID = "iniID",
  retDataFrame = TRUE,
  callFrom = NULL,
  silent = FALSE,
  debug = FALSE
)
matchNamesWithReverseParts

Arguments

- **dat** (matrix or data.frame) main input for making non-redundant
- **summarizeRedAs** (character) summarization method(s), typical choices 'median', 'mean', 'min' or 'maxOfRef', 'maxAbsOfRef' for summarizing according to 1 specified column, may be single method for all or different method for each column (besides col 'iniID') or special method looking at column (if found, first of special methods used, everything else not considered).
- **iniID** (character) column-name used as initial ID (default="iniID")
- **retDataFrame** (logical) if TRUE, check if text-columns may be converted to data.frame with numeric
- **callFrom** (character) allows easier tracking of message(s) produced
- **silent** (logical) suppress messages
- **debug** (logical) for bug-tracking: more/enhanced messages

Value

(numeric) matrix or data.frame with summarized data and add'l col with number of initial redundant lines

See Also

- simple/partial functionality in `summarizeCols`, `checkSimValueInSer`

Examples

```r
  t3 <- data.frame(ref=rep(11:15,3),tx=letters[1:15],
                   matrix(round(runif(30,-3,2),1),nc=2),stringsAsFactors=FALSE)
by(t3,t3[,1],function(x) x)
(sapply(by(t3,t3[,1],function(x) x), summarizeCols, me="maxAbsOfRef"))
(xt3 <- makeNRedMatr(t3, summ="mean", iniID="ref"))
(xt3 <- makeNRedMatr(t3, summ=unlist(list(X1="maxAbsOfRef")), iniID="ref"))
```

Description

This function provides a variant to `match`, where initially non-matching elements of `x` will be tested by decomposing non-matching elements, reversing the parts in front and after the separator `sep` and re-matching. If separator `sep` does not occur, a warning will be issued, if it occurs more than once, the parts before and after the first separator will be used and a warning issued.

Usage

```r
matchNamesWithReverseParts(x, y, sep = "-", silent = FALSE, callFrom = NULL)
```
matr2list

Transform columns of matrix to list of vectors

Description
convert matrix to list of vectors: each column of 'mat' as vector of list

Usage
matr2list(mat, concSym = ".", silent = FALSE, callFrom = NULL)

Arguments
mat (matrix) main input
concSym (character) symbol for concatenating: concatenation of named vectors in list names as colname(s)+'concSym'+rownname
silent (logical) suppress messages
callFrom (character) allow easier tracking of messages produced

Value
matrix or array (1st dim is intraplate-position, 2nd .. plate-group/type, 3rd .. channels)

Arguments
x (character) first vector for match
y (character) second vector for match
sep (character) separator between elements
silent (logical) suppress messages
callFrom (character) allow easier tracking of message(s) produced

Value
index for matching (integer) x to y

See Also
match

Examples
tx1 <- c("a-b","a-c","d-a","d-b","b-c","d-c")
tmp <- triCoord(4)
tx2 <- paste(letters[tmp[,1]],letters[tmp[,2]],sep="-")
## Some matches won't be found, since 'a-d' got reversed to 'd-a', etc...
match(tx1,tx1)
matchNamesWithReverseParts(tx1,tx2)
mergeSelCol

See Also
convToNum

Examples
mat1 <- matrix(1:12,ncol=3,dimnames=list(letters[1:4], LETTERS[1:3]))
mat2 <- matrix(LETTERS[11:22], ncol=3,dimnames=list(letters[1:4], LETTERS[1:3]))
matr2list(mat1); matr2list(mat2)

mergeSelCol  Merge selected columns out of 2 matrix or data.frames

Description
mergeSelCol merges selected columns out of 2 matrix or data.frames. 'selCols' will be used to define columns to be used; optionally may be different for 'dat2' : define in 'supCols2'. Output-cols will get additions specified in newSuff (default '.x' and '.y')

Usage
mergeSelCol(
dat1,
dat2,
selCols,
supCols2 = NULL,
byC = NULL,
useAll = FALSE,
setRownames = TRUE,
newSuff = c(".x", ".y"),
callFrom = NULL
)

Arguments
dat1  matrix or data.frame for fusing
dat2  matrix or data.frame for fusing
selCols  will be used to define columns to be used; optionally may be different for 'dat2' : define in 'supCols2'
supCols2  if additional column-names should be extracted form dat2
byC  (character) 'by' value used in merge
useAll  (logical) use all lines (will produce NAs when given identifier not found un 2nd group of data)
setRownames  (logical) if TRUE, will use values of col used as 'by' as rownames instead of showing as add’l col in output
newSuff  (character) prefix (argument 'suffixes' in merge)
callFrom  (character) allow easier tracking of message(s) produced
Value
data.frame

See Also
merge, merge 3 data.frames using mergeSelCol3

Examples
mat1 <- matrix(c(1:7, letters[1:7], 11:17), ncol=3, dimnames=list(LETTERS[1:7], c("x1", "x2", "x3")))
mat2 <- matrix(c(1:6, c("b", "a", "e", "f", "g", "k"), 31:36),
ncol=3, dimnames=list(LETTERS[11:16], c("y1", "x2", "x3")))
mergeSelCol(mat1, mat2, selC=c("x2", "x3"))

mergeSelCol3

Description
successive merge of selected columns out of 3 matrix or data.frames. 'selCols' will be used to define columns to be used; optionally may be different for 'dat2': define in 'supCols2'. Output-cols will get additions specified in newSuff (default '.x' and '.y')

Usage
mergeSelCol3(
  dat1,
  dat2,
  dat3,
  selCols,
  supCols2 = NULL,
  supCols3 = NULL,
  byC = NULL,
  useAll = FALSE,
  setRownames = TRUE,
  newSuff = c(".x", ".y", ".z"),
  callFrom = NULL
)

Arguments
dat1 matrix or data.frame for fusing
dat2 matrix or data.frame for fusing
dat3 matrix or data.frame for fusing
selCols will be used to define columns to be used; optionally may be different for 'dat2': define in 'supCols2'
mergeW2

supCols2 if additional column-names should be extracted form dat2
supCols3 if additional column-names should be extracted form dat3
byC (character) 'by' value used in merge
useAll (logical) use all lines (will produce NAs when given identifier not found in 2nd group of data)
setRownames if TRUE, will use values of col used as 'by' as rownames instead of showing as add'l col in output
newSuff (character) prefix (argument 'suffixes' in merge)
callFrom (character) allow easier tracking of message(s) produced

Value
data.frame

See Also
merge, mergeSelCol

Examples

mat1 <- matrix(c(1:7,letters[1:7],11:17),ncol=3,dimnames=list(LETTERS[1:7],c("x1","x2","x3")))
mat2 <- matrix(c(1:6,c("b","a","e","f","g","k"),31:36),ncol=3,
dimnames=list(LETTERS[11:16],c("y1","x2","x3")))
mat3 <- matrix(c(1:6,c("c","a","e","b","g","k"),51:56),ncol=3,
dimnames=list(LETTERS[11:16],c("z1","x2","x3")))
mergeSelCol3(mat1,mat2,mat3,selC=c("x2","x3"))

mergeW2

Extended version of merge for multiple objects (even without rownames)

Description

mergeW2 provides flexible merging out of 'MArrayLM'-object (if found, won't consider any other input-data) or of separate vectors or matrices. The main idea was to have something not adding add'l lines as merge might do, but to stay within the frame of the 1st argument given, even when IDs are repeated, so the output follows the order of the 1st argument, non-redundant IDs are created (orig IDs as new column). If no 'MArrayLM'-object found: try to combine all elements of input '...', input-names must match predefined variants 'chInp'. IDs given in 1st argument and not found in later arguments will be displayed as NA in the output matrix of data.frame. Note : (non-data) arguments must be given with full name (so far no lazy evaluation, may conflict with names in 'inputNamesLst'). Note : special characters in colnames bound to give trouble. Note : when no names given, mergeW2 will presume order of elements (names) from 'inputNamesLst'. PROBLEM : error after xxMerg3 when several entries have matching (row)names but some entries match only partially (what to do : replace with NAs ??)
Usage
mergeW2(
  ...,
  nonRedundID = TRUE,
  convertDF = TRUE,
  selMerg = TRUE,
  inputNamesLst = NULL,
  noMatchPursue = TRUE,
  standColNa = FALSE,
  lastOfMultCols = c("p.value", "Lfdr"),
  duplTxtSep = "_",
  silent = FALSE,
  debug = FALSE,
  callFrom = NULL
)

Arguments
... all data (vectors, matrices or data.frames) intendes for merge
nonRedundID (logical) if TRUE, allways add 1st column with non-redundant IDs (add anyway
if non-redundant IDs found )
convertDF (logical) allows converting output in data.frame, add new heading col with non-
red rownames & check which cols should be numeric
selMerg (logical) if FALSE toggle to classic merge() (will give more rows in output in
case of redundant names
inputNamesLst (list) named list with character vectors (should be unique), search these names
in input for extracting/merging elements use for 'lazy matching' when checking
names of input, default: 7 groups ('Mvalue', 'Avalue', 'p.value', 'mouseInfo', 'Lfdr', 'link', 'filt')
with common short versions
noMatchPursue (logical) allows using entries where 0 names match (just as if no names given)
standColNa (logical) if TRUE return standard colnames as defined in 'inputNamesLst' (ie
'chInp'), otherwise colnames as initially provided
lastOfMultCols may specify input groups where only last col will be used/extracted
duplTxtSep (character) separator for counting/denomiating multiple occurances of same name
silent (logical) suppress messages
debug (logical) for bug-tracking: more/enhanced messages and intermediate objects
written in global name-space
callFrom (character) allows easier tracking of message(s) produced

Value
matrix or data.frame of fused data

See Also
merge
**Examples**

```r
t1 <- 1:10; names(t1) <- letters[c(1:7,3:4,8)]
t2 <- 20:11; names(t2) <- letters[c(1:7,3:4,8)]
t3 <- 101:110; names(t3) <- letters[c(11:20)]
t4 <- matrix(100:81,ncol=2,dimnames=list(letters[1:10],c("co1","co2")))
t5 <- cbind(t1=t1,t52=t1+20,t53=t1+30)
t1; t2; t3; cbind(t1,t2)
mergeW2(Mval=t1,p.value=t2,debug=FALSE)
```

---

### minDiff

**Minimum distance/difference between values**

**Description**

minDiff aims to find the min distance (ie closest point) to any other x (numeric value), ie intra 'x' and returns matrix with 'index','value','dif','ppm','ncur','nbest','best'. At equal distance to lower & upper neighbour point, the upper (following) point is chosen (as single best). In case of multiple ex-aequo distance returns 1st of multiple, may be different at various repeats.

**Usage**

```r
minDiff(x, digSig = 3, ppm = TRUE, initOrder = TRUE, callFrom = NULL)
```

**Arguments**

- `x` (numeric) vector to search minimum difference
- `digSig` number of significant digits, used for ratio or ppm column
- `ppm` (logical) display distance as ppm (1e6*diff/refValue, ie normalized difference eg as used in mass spectrometry), otherwise the ratio is given as : value(from 'x') / closestValue (from 'x')
- `initOrder` (logical) return matrix so that 'x' matches exactly 2nd col of output
- `callFrom` (character) allow easier tracking of message(s) produced

**Value**

matrix

**See Also**

dist

**Examples**

```r
set.seed(2017); aa <- 100*c(0.1+round(runif(20),2),0.53,0.53)
minDiff(aa); minDiff(aa,initO=TRUE,ppm=FALSE); .minDif(unique(aa))
```
moderTest2grp

Moderated pair-wise t-test from limma

Description

Runs moderated t-test from package 'limma' on each line of data. Note: This function requires the package limma from bioconductor. The limma contrast-matrix has to be read by column, the lines in the contrast-matrix containing '+1' will be compared to the '-1' lines, eg grpA-grpB.

Usage

moderTest2grp(
  dat,
  grp,
  limmaOutput = TRUE,
  addResults = c("lfdr", "FDR", "Mval", "means"),
  testOrientation = "=",
  silent = FALSE,
  callFrom = NULL
)

Arguments

dat           matrix or data.frame with rows for multiple (independent) tests, use ONLY with 2 groups; assumed as log2-data
grp           (factor) describes column-relationship of 'dat' (1st factor is considered as reference -> orientation of M-values !!)
limmaOutput   (logical) return full (or extended) MArrayLM-object from limma or 'FALSE' for only the (uncorrected) p.values
addResults    (character) types of results to add besides basic limma-output (eg "lfdr" using fdrtools-package,"FDR" for BY-FDR,"Mval" (assumes that data are log2 !),"means" or "nonMod" for non-moderated test)
testOrientation    (character) for one-sided test (">","greater" or "<","less"), NOTE : 2nd grp is considered control/reference, '<' will identify grp1 < grp2
silent         (logical) suppress messages
callFrom       (character) allow easier tracking of message(s) produced

Value

limma-type MA-object (list)

See Also

lmFit and eBayes


**Examples**

```r
set.seed(2017); t8 <- matrix(round(rnorm(1600,10,0.4),2),ncol=8,
  dimnames=list(paste("l",1:200),c("AA1","BB1","CC1","DD1","AA2","BB2","CC2","DD2")))
t8[3:6,1:2] <- t8[3:6,1:2]+3 # augment lines 3:6 for AA1&BB1
t8[5:8,5:6] <- t8[5:8,5:6]+3 # augment lines 5:8 for AA2&BB2 (c,d,g,h should be found)
t4 <- log2(t8[,1:4]/t8[,5:8])
fit4 <- moderTest2grp(t4,gl(2,2))
limma::topTable(fit4,coef=1,n=5) # effect for 3,4,7,8
fit4in <- moderTest2grp(t4,gl(2,2),testO="<")
limma::topTable(fit4in,coef=1,n=5)
```

---

**moderTestXgrp**  
*Multiple moderated pair-wise t-tests from limma*

**Description**

Runs all pair-wise combinations of moderated t-tests from package 'limma' on each line of data against 1st group from 'grp'. Note: This function requires the package limma from bioconductor. The limma contrast-matrix has to be read by column, the lines in the contrast-matrix containing '+1' will be compared to the '-1' lines, eg grpA-grpB.

**Usage**

```r
moderTestXgrp(
  dat, 
  grp, 
  limmaOutput = TRUE, 
  addResults = c("lfdr", "FDR", "Mval", "means"), 
  testOrientation = "=", 
  silent = FALSE, 
  callFrom = NULL 
)
```

**Arguments**

- **dat**  
  matrix or data.frame with rows for multiple (independent) tests, use ONLY with 2 groups; assumed as log2-data !!!

- **grp**  
  (factor) describes column-relationship of 'dat' (1st factor is considered as reference -> orientation of M-values !!)

- **limmaOutput**  
  (logical) return full (or extended) MArrayLM-object from limma or 'FALSE' for only the (uncorrected) p.values

- **addResults**  
  (character) types of results to add besides basic limma-output (eg "lfdr" using fdrtools-package,"FDR" for BY-FDR,"Mval" (assumes that data are log2 !),"means" or "nonMod" for non-moderated test)

- **testOrientation**  
  (character) for one-sided test (">","greater" or "<","less"), NOTE : 2nd grp is considered control/reference, '<' will identify grp1 < grp2
silent (logical) suppress messages
callFrom (character) allow easier tracking of message(s) produced

Value

limma-type MA-object (list)

See Also
eBayes for basal tool and moderTest2grp for single comparison

Examples

gp <- factor(rep(LETTERS[c(3,1,4)],c(2,3,3)))
set.seed(2017); t8 <- matrix(round(rnorm(208*8,10,0.4),2),ncol=8,
dimnames=list(paste(letters[],rep(1:8,each=26),sep=""),paste(grp,c(1:2,1:3,1:3),sep="")))
t8[3:6,1:2] <- t8[3:6,1:2] +3  # augment lines 3:6 (c-f)
t8[5:8,c(1:2,6:8)] <- t8[5:8,c(1:2,6:8)] -1.5  # lower lines
t8[6:7,3:5] <- t8[6:7,3:5] +2.2  # augment lines
## expect to find C/A in c,d,g, (h)
## expect to find C/D in c,d,e,f
## expect to find A/D in f,g,(h)
test8 <- moderTestXgrp(t8,grp)
head(test8$p.value,n=8)

Description

naOmit removes NAs from input vector. This function has no slot for removed elements while na.omit does so. Resulting objects from naOmit are smaller in size and subsequent execution (on large vectors) is faster (in particular if many NAs get encountered). Note: Behaves differently to na.omit with input other than plain vectors. Will not work with data.frames!

Usage

naOmit(x)

Arguments

x (vector or matrix) input

Value

vector without NAs (matrix input will be transformed to vector). Returns NULL if input consists only of NAs.
nFragments

See Also

na.fail, na.omit

Examples

aA <- c(11:13,NA,10,NA);
naOmit(aA)

nFragments

Number of fragments after cut at specific character(s) within size-range

Description

nFragments determines number of fragments /entry within range of 'sizeRa' (numeric,length=2) when cutting after 'cutAt'

Usage

nFragments(protSeq, cutAt, sizeRa)

Arguments

protSeq (character) text to be cut

cutAt (character) position to cut

sizeRa (numeric,length=2) min and max size to consider

Value

numeric vector with number of fragments for each entry 'protSeq' (names are 'protSeq')

See Also

cutAtMultSites, simple version {nFragments0} (no size-range)

Examples

tmp <- "MSVSREDSCELDLVYVTERIAAVSFPMSTANEENFRSNLREVAQMLKSKHGNLFFNLSERRPDITKHLAKVKLFNPDGLHTPALEKI"
nFragments(c(tmp,"ojoRij"),c("R","K"),c(4,31))
**nFragments0**

*Number of fragments after cut at specific character(s)*

**Description**

nFragments0 tells the number of fragments/entry when cutting after 'cutAt'

**Usage**

nFragments0(protSeq, cutAt)

**Arguments**

- **protSeq** (character) text to be cut
- **cutAt** (integer) position to cut

**Value**

numeric vector with number of fragments for each entry 'protSeq' (names are 'protSeq')

**See Also**

more elaborate {nFragments}; cutAtMultSites

**Examples**

tmp <- "MSVSRMedSCELDMIVYTHERIAVFEPSTANENFRSRLREVAQMLKSKHGNLLFNLSERPPDTKLHAKNLGEPDLHTPALEK"
nFragments0(c(tmp,"ojioRij"),c("R","K"))

**nNonNumChar**

*Count number of non-numeric characters*

**Description**

nNonNumChar counts number of non-numeric characters. Made for positive non-scientific values (eg won’t count neg-sign, neither Euro comma ",")

**Usage**

nNonNumChar(txt)

**Arguments**

- **txt** character vector to be treated
Value

numeric vector with numer of non-numeric characters (ie not '.' or 0-9))

See Also

nchar

Examples

nNonNumChar("a1b"); sapply(c("aa","12ab","a1b2","12","0.5"),nNonNumChar)

nonAmbiguousMat

Transform matrix to non-ambiguous matrix (in respect to given column)

Description

nonAmbiguousMat makes values of matrix 'mat' in col 'byCol' unique.

Usage

nonAmbiguousMat(
    mat, byCol, uniqOnly = FALSE, asList = FALSE, nameMod = "amb_", callFrom = NULL
)

Arguments

mat numeric or character matrix (or data.frame), column specified by 'byCol' must be/will be used as numeric, 1st column of 'mat' will be considered like index & used for adding prefix 'nameMod' (unless byCol=1, then 2nd col will be used)

byCol (character or integer-index) column by which ambiguousity will be tested

uniqOnly (logical) if =TRUE return unique only, if =FALSE return unique and single representative of non-unique values (with " added to name), selection of representative of repeated: first (of sorted) or middle if >2 instances

asList (logical) return result as list

nameMod (character) prefix added to 1st column of 'mat' (expect 'by') for indicating non-unique/ambiguous values

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

Value

sorted non-ambiguous numeric vector (or list if 'asList'=TRUE and 'uniqOnly'=FALSE)
nonAmbiguousNum

See Also

for non-numeric use firstOfRepeated - but 1000x much slower !: get1stOfRepeatedByCol

Examples

set.seed(2017): mat2 <- matrix(c(1:100,round(rnorm(200),2)),ncol=3,
  dimnames=list(1:100,LETTERS[1:3]));
head(mat2U <- nonAmbiguousMat(mat2,by="B",na="_",uniqO=FALSE),n=15)
head(get1stOfRepeatedByCol(mat2,sortB="B",sortS="B"))

nonAmbiguousNum make numeric vector non-ambiguous (ie unique)

Description

nonAmbiguousNum makes (named) values of numeric vector 'x' unique. Note: for non-numeric use firstOfRepeated - but 1000x slower ! Return sorted non-ambiguous numeric vector (or list if 'asList'=TRUE and 'uniqOnly'=FALSE)

Usage

nonAmbiguousNum(
  x,
  uniqOnly = FALSE,
  asList = FALSE,
  nameMod = "amb_",
  callFrom = NULL
)

Arguments

  x  (numeric) main input
  uniqOnly  (logical) if=TRUE return unique only, if =FALSE return unique and single representative of non-unique values (with " added to name), selection of representative of repeated: first (of sorted) or middle if >2 instances
  asList  (logical) return list
  nameMod  (character) text to add in case on ambiguous values, default="amb_"
  callFrom  (character) allow easier tracking of message(s) produced

Value

  sorted non-ambiguous numeric vector (or list if 'asList'=TRUE and 'uniqOnly'=FALSE)

See Also

  firstOfRepeated for non-numeric use (much slower !!), duplicated
nonredDataFrame

Examples

set.seed(2017); aa <- round(rnorm(100),2); names(aa) <- 1:length(aa)
str(nonAmbiguousNum(aa))
str(nonAmbiguousNum(aa, uniq=FALSE, asLi=TRUE))

-----

nonredDataFrame

Filter for unique elements

Description

nonredDataFrame filters 'x' (list of char-vectors or char-vector) for elements unique (to 'ref' or if NULL to all 'x') and of character length. May be used for different 'accession' for same pep sequence (same 'peptide_id'). Note: made for treating data.frames, may be slightly slower than matrix equivalent

Usage

nonredDataFrame(
  dataFr,
  useCol = c(pepID = "peptide_id", protID = "accession", seq = "sequence", mod = "modifications"),
  sepCollapse = "//",
  callFrom = NULL
)

Arguments

dataFr (data.frame) main input

useCol (character,length=2) column names of 'dataFr' to use: 1st value designates where redundant values should be gathered; 2nd value designates column of which information should be concatenated

sepCollapse (character) concatenation symbol

callFrom (character) allow easier tracking of messages produced

Value

data.frame of filtered (fewer lines) with additional 2 columns 'nSamePep' (number of redundant entries) and 'concID' (concatenated content)

See Also

combineRedBasedOnCol, correctToUnique, unique

Examples

df1 <- data.frame(cbind(xA=letters[1:5],xB=c("h","h","f","e","f"),xC=LETTERS[1:5]))
nonredDataFrame(df1,useCol=c("xB","xC"))
nonRedundLines  Non-redundant lines of matrix

Description

nonRedundLines reduces complexity of matrix (or data.frame) if multiple consecutive (!) lines with same values. Return matrix (or data.frame) without repeated lines (keep 1st occurance)

Usage

nonRedundLines(dat, callFrom = NULL)

Arguments

dat  (matrix or data.frame) main input
callFrom  (character) allow easier tracking of message(s) produced

Value

matrix (or data.frame) without repeated lines (keep 1st occurance).

See Also

firstLineOfDat, firstOfRepLines, findRepeated, firstOfRepeated, get1stOfRepeatedByCol, combineRedBasedOnCol, correctToUnique

Examples

mat2 <- matrix(rep(c(1,1:3,3,1),2),ncol=2,dimnames=list(letters[1:6],LETTERS[1:2]))
nonRedundLines(mat2)

normalizeThis  Normalize data in various modes

Description

Generic normalization of 'dat' (by columns), multiple methods may be applied. The choice of normalization procedures must be done with care, plotting the data before and after normalization may be critical to understanding the initial data structure and the effect of the procedure applied. Inappropriate methods chosen may render interpretation of (further) results incorrect. Normalization using the method vsn runs justvsn from vsn (this requires a minimum of 42 rows of input-data). Note: Depending on the procedure chosen, the normalized data may appear on a different scale.
**normalizeThis**

**Usage**

```r
normalizeThis(
  dat, 
  method = "mean", 
  refLines = NULL, 
  refGrp = NULL, 
  trimFa = NULL, 
  quantFa = NULL, 
  expFa = NULL, 
  silent = FALSE, 
  callFrom = NULL
)
```

**Arguments**

- **dat**: matrix or data.frame
- **method**: (character) may be "mean", "median", "NULL", "none", "trimMean", "slope", "exponent", "slope2Sections", "vsn"; When NULL or 'none' is chosen the input will be returned
- **refLines**: (NULL or numeric) allows to consider only specific lines of 'dat' when determining normalization factors (all data will be normalized)
- **refGrp**: Only the columns indicated will be used as reference, default all columns (integer or colnames)
- **trimFa**: (numeric, length=1) additional parameters for trimmed mean
- **quantFa**: (numeric, length=2) additional parameters for quantiles to use with method='slope'
- **expFa**: (numeric, length=1) additional parameters for method='exponent'
- **silent**: (logical) suppress messages
- **callFrom**: (character) allow easier tracking of message produced

**Value**

matrix of normalized data

**See Also**

exponNormalize, adjBy2ptReg, justvsn

**Examples**

```r
set.seed(2015); rand1 <- round(runif(300)+rnorm(300,0,2),3)
dat1 <- cbind(ser1=round(100:1+rand1[1:100]),ser2=round(1.2*(100:1+rand1[101:200])-2),
  ser3=round((100:1+rand1[201:300])^1.2-3))
dat1[,dat1 <1] <- NA
summary(dat1)
head(.normalize(dat1,"mean",list()))
dat1[c(1:5,50:54,95:100),]
```
organizeAsListOfRepl

(re)organize data of (3-dim) array as list of replicates

Description

Organize array of all data ('arrIn', long table) into list of (replicate-)arrays (of similar type/layout) based on dimension number 'byDim' of 'arrIn' (eg 2nd or 3rd dim). Argument inspNChar defines the number of characters to consider, so if the beginning of names is the same they will be separated as list of multiple arrays. Default will search for '_' separator or trim from end if not found in the relevant dimnames

Usage

organizeAsListOfRepl(
  arrIn, 
  inspNChar = 0, 
  byDim = 3, 
  silent = TRUE, 
  callFrom = NULL 
)

Arguments

arrIn (array) main input

inspNChar (integer) if inspNChar=0 the array-names (2nd dim of 'arrIn') will be cut before last '_'

byDim (integer, length=1) dimension number along which data will be split in separate elements (considering the first inspNChar characters)

silent (logical) suppress messages

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

Value

list of arrays (typically 1st and 2nd dim for specific genes/objects, 3rd for different measures associated with)
**Examples**

```r
arr1 <- array(1:24,dim=c(4,3,2),dimnames=list(c(LETTERS[1:4]),
paste("col",1:3,sep=""),c("ch1","ch2")))
organizeAsListOfRepl(arr1)
```

**Description**

`partialDist` calculates distance matrix like `dist` for 1- or 2-dim data, but only partially, ie only cases of small distances. This function was made for treating very large data-sets where only very close distances to a given point need to be found, it allows to overcome memory-problems with larger data (and faster execution with > 50 rows of `dat`).

**Usage**

```r
partialDist(
  dat,
  groups,
  overLap = TRUE,
  method = "euclidean",
  silent = FALSE,
  callFrom = NULL
)
```

**Arguments**

- `dat` (matrix of numeric values) main input
- `groups` (factor) to split using `cut` or specific custom grouping (length of `dat`)
- `overLap` (logical) if TRUE make groups overlapping by 1 value (ie maintain some context-information)
- `method` 'character' name of method passed to `dist`
- `silent` (logical) suppres messages
- `callFrom` (character) allow easier tracking of message(s) produced

**Value**

matrix (not of class 'dist')

**See Also**

`dist`
Examples

```r
set.seed(2016); mat3 <- matrix(runif(300),nr=30)
round(dist(mat3),1)
round(partialDist(mat3,gr=3),1)
```

partUnlist

**Partial unlist of lists of lists**

Description

partUnlist does partial unlist for treating list of lists: New (returned) list has one level less of hierarchy (Highest level list will be appended). In case of conflicting (non-null) listnames a prefix will be added. Behaviour different to `unlist` when unlisting list of matrices.

Usage

```r
partUnlist(lst)
```

Arguments

- `lst` list to be partially unlisted

Value

list with partially reduced nested structure

See Also

`unlist`, `asSepList`

Examples

```r
partUnlist(list(list(a=11:12,b=21:24),list(c=101:101,d=201:204)))
li4 <- list(c=1:3,L2=matrix(1:4,ncol=2),li3=list(L1=11:12,L2=matrix(21:26,ncol=2)))
partUnlist(li4)
unlist(li4,rec=FALSE)
```
**Description**

`pasteC` is a variant of `paste` for convenient use of paste-collapse and separation of last element to paste (via `lastCol`). This function was made for more human-like enumeration in output and messages. If multiple arguments are given without names they will all be concatenated, if they contain names lazy evaluation for names will be tried (with preference to longest match to argument names). Note that some special characters (like backslash) may need to be protected when used with `collapse` or `quoteC`. Returns character vector of length 1 (everything pasted together).

**Usage**

```r
pasteC(..., collapse = '', lastCol = " and ", quoteC = "")
```

**Arguments**

- `...` (character) main input to be collapsed
- `collapse` (character, length=1) element to use for collapsing
- `lastCol` (character) text to use before last item enumerated element
- `quoteC` (character) character to use for citing with quotations (default"")

**Value**

character vector of length=1 of the concatenated input/values.

**See Also**

- `paste` for basic paste

**Examples**

```r
pasteC(1:4)
```

---

**Description**

`presenceFilt` produces logical matrix to be used as filter for lines of `dat` for sufficient presence of non-NA values (ie limit number of NAs per line). Filter abundance-expression data for min number and/or ratio of non-NA values in at least 1 of multiple groups. This type of procedure is common in proteomics and transcriptomics, where a NA can many times be associated with quantitation below detection limit.

**Description**

`presenceFilt` produces logical matrix to be used as filter for lines of `dat` for sufficient presence of non-NA values (ie limit number of NAs per line). Filter abundance-expression data for min number and/or ratio of non-NA values in at least 1 of multiple groups. This type of procedure is common in proteomics and transcriptomics, where a NA can many times be associated with quantitation below detection limit.

**Description**

`presenceFilt` produces logical matrix to be used as filter for lines of `dat` for sufficient presence of non-NA values (ie limit number of NAs per line). Filter abundance-expression data for min number and/or ratio of non-NA values in at least 1 of multiple groups. This type of procedure is common in proteomics and transcriptomics, where a NA can many times be associated with quantitation below detection limit.

**Description**

`presenceFilt` produces logical matrix to be used as filter for lines of `dat` for sufficient presence of non-NA values (ie limit number of NAs per line). Filter abundance-expression data for min number and/or ratio of non-NA values in at least 1 of multiple groups. This type of procedure is common in proteomics and transcriptomics, where a NA can many times be associated with quantitation below detection limit.
Usage

```r
presenceFilt(
  dat,
  grp,
  maxGrpMiss = 1,
  ratMaxNA = 0.8,
  minVal = NULL,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

- **dat**: matrix or data.frame (abundance or expression-values which may contain some NAs).
- **grp**: factor of min 2 levels describing which column of 'dat' belongs to which group (levels 1 & 2 will be used)
- **maxGrpMiss**: (numeric) at least 1 group has not more than this number of NAs (otherwise marke line as bad)
- **ratMaxNA**: (numeric) at least 1 group reaches this content of non-NA values
- **minVal**: (default NULL or numeric), any value below will be treated like NA
- **silent**: (logical) suppress messages
- **callFrom**: (character) allow easier tracking of message produced

Value

logical matrix (with separate col for each pairwise combination of 'grp' levels) indicating if line of 'dat' acceptable based on NAs (and values minVal)

Examples

```r
dat1 <- matrix(1:56, ncol=7)
dat1[c(2,3,4,5,6,10,12,18,20,22,23,26,27,28,30,31,34,38,39,50,54)] <- NA
dat1; presenceFilt(dat1, gr=gl(3,3)[-c(3:4)], maxGr=0)
presenceFilt(dat1, gr=gl(2,4)[-1], maxGr=1, ratM=0.1)
presenceFilt(dat1, gr=gl(2,4)[-1], maxGr=2, rat=0.5)
```

---

**pVal2lfdr**

*Convert p-values to lfdr*

Description

This function takes a numeric vector of p-values and returns a vector of lfdr-values (local false discovery) using the package *fdrtool*. Multiple testing correction should be performed with caution, short series of p-values typically pose problems for transforming to lfdr. The transformation to lfdr values may give warning messages, in this case the resultant lfdr values may be invalid!
Usage

   pVal2lfdr(x, silent = TRUE, callFrom = NULL)

Arguments

   x  (numeric) vector of p.values
   silent  (logical) suppres messages
   callFrom  (character) allow easier tracking of message(s) produced

Value

   (numeric) vector of lfdr values (or NULL if data insufficient to run the function 'fdrtool')

See Also

   lfdr from fdrtool, other p-adjustments (multiple test correction, eg FDR) in p.adjust

Examples

   ## Note that this example is too small for estimating really meaningful fdr values
   set.seed(2017); t8 <- matrix(round(rnorm(160,10,0.4),2),ncol=8,dimnames=letter[c(1:20, c("AA1","BB1","CC1","DD1","AA2","BB2","CC2","DD2"))]
   t8[3:6,1:2] <- t8[3:6,1:2]+3  # augment lines 3:6 (c-f) for AA1&BB1
   t8[5:8,5:6] <- t8[5:8,5:6]+3  # augment lines 5:8 (e-h) for AA2&BB2 (c,d,g,h should be found)
   head(pVal2lfdr(apply(t8,1,function(x) t.test(x[1:4],x[5:8])$p.value)))

---

randIndFx

Distance of categorical data (Jaccard,Rand and adjusted Rand index)

Description

   randIndFx calculates distance of categorical data (as Rand Index, Adjusted Rand Index or Jaccard Index). Note: uses/requires package flexclust Methods so far available (via flexclust): "ARI" .. adjusted Rand Index, "RI" .. Rand index, "J" .. Jaccard, "FM" .. Fowlkes-Mallows.

Usage

   randIndFx(ma, method = "ARI", adjSense = TRUE, silent = FALSE, callFrom = NULL)

Arguments

   ma  (matrix) main input for distance calculation
   method  (character) name of distance method (eg "ARI","RI","J","FM")
   adjSense  (logical) allows introducing correlation/anticorrelation (interprete neg distance results as anti)
   silent  (logical) suppres messages
   callFrom  (character) allow easier tracking of message(s) produced
Value

distance matrix

See Also

comPart in randIndex

Examples

set.seed(2016); tab2 <- matrix(sample(1:2,size=42,replace=TRUE),ncol=7)
flexclust::comPart(tab2[1,],tab2[2,])
flexclust::comPart(tab2[1,],tab2[3,])
flexclust::comPart(tab2[1,],tab2[4,])
  randIndFx(tab2,adjS=FALSE)
cor(t(tab2))
  randIndFx(tab2,adjS=TRUE)

Description

Count the number of instances where the corresponding columns of 'dat' have a value matching
the group number as specified by 'grp'. Counting will be performed/repeated independently for
each line of 'dat'. Returns array (1st dim is rows of dat, 2nd is unique(grp), 3rd dim is ok/bad),
these results may be tested using eg fisher.test. This function was made for prearing to test the
ranking of multiple features (lines in 'mat') including replicates (levels of 'grp').

Usage

rankToContigTab(dat, grp)

Arguments

dat (matrix or data.frame of integer values) ranking of multiple features (lines),
equal ranks may occur

grp (integer) expected ranking

Value

array (1st dim is rows of dat, 2nd is unique(grp), 3rd dim is ok/bad)

See Also

lm
Examples

Let's create a matrix with ranks (equal ranks do occur)

```r
ma0 <- matrix(rep(1:3,each=6),ncol=6,dimnames=list(c("li1","li2","ref"),letters[1:6]))
ma0[1,6] <- 1  # create item not matching correctly
ma0[2,] <- c(3:1,2,1,3)  # create items not matching correctly
gr0 <- gl(3,2)  # the expected ranking (as duplicates)
(count0 <- rankToContigTab(ma0,gr0))
cTab <- t(apply(count0,c(1,3),sum))
```

Now we can compare the ranking of line1 to ref ... 

```r
fisher.test(cTab[,c(3,1)])  # test li1 against ref
disher.test(cTab[,c(3,2)])  # test li2 against ref
```

---

**ratioAllComb**

*Calculate all ratios between x and y*

**Description**

ratioAllComb calculates all possible pairwise ratios between all individual values of x and y.

**Usage**

```r
ratioAllComb(x, y, maxLim = 10000, silent = FALSE, callFrom = NULL)
```

**Arguments**

- `x` (numeric) vector, numerator for constructing ratios
- `y` (numeric) vector, denominator for constructing ratios
- `maxLim` (integer) allows reducing complexity by drawing for very long x or y
- `silent` (logical) suppress (less important) messages
- `callFrom` (character) allow easier tracking of message(s) produced

**Value**

(numeric) vector with all ratios

**Examples**

```r
set.seed(2014); ra1 <- c(rnorm(9,2,1),runif(8,1,2))
ratioAllComb(ra1[1:9],ra1[10:17])
boxplot(list(norm=ra1[1:9],unif=ra1[10:17],rat=ratioAllComb(ra1[1:9],ra1[10:17])))
```
**ratioToPpm**  
*Convert ratio to ppm*

**Description**

`ratioToPpm` transforms ratio 'x' to ppm (parts per million). If 'y' not given (or different length as 'x'), then 'x' is assumed as ratio otherwise ratios are constructed as x/y is used lateron. Does additional checking: negative values not expected - will be made absolute!

**Usage**

```r
ratioToPpm(x, y = NULL, nSign = NULL, silent = FALSE, callFrom = NULL)
```

**Arguments**

- **x** (numeric) main input
- **y** (numeric) optional value to construct ratios (x/y). If NULL (or different length as 'x'), then 'x' will be considered as ratio.
- **nSign** (numeric) number of significant digits
- **silent** (logical) suppress messages
- **callFrom** (character) allow easier tracking of message(s) produced

**Value**

numeric vector of ppm values

**See Also**

`XYToDiffPpm` for ppm of difference as used in mass spectrometry

**Examples**

```r
set.seed(2017); aa <- c(1.000001, 0.999999, 1 + rnorm(10, 0, 0.001))
cbind(x = aa, ppm = ratioToPpm(aa, nSign = 4))
```
readCsvBatch  

Read batch of csv-files

Description

This function was designed to read screening data split in parts (with common structure) and saved to multiple files, to extract the numeric columns and to compile all (numeric) data to a single array (or list). Some screening platforms save results while progressing through a pile of microtiter-plates separately. The organization of the resultant files is structured through file-names and all files have exactly the same organization of lines and columns/ European or US-formatted csv files can be read, if argument fileFormat is NULL both types will be tested, otherwise it allows to specify a given format. The presence of headers (to be used as column-names) may be tested using checkFormat.

Usage

readCsvBatch(
  fileNames = NULL,
  path = ".",
  fileFormat = "Eur",
  checkFormat = TRUE,
  returnArray = TRUE,
  columns = c("Plate", "Well", "StainA"),
  excludeFiles = "All infected plates",
  simpleNames = TRUE,
  minNamesLe = 4,
  silent = FALSE,
  callFrom = NULL
)

Arguments

fileNames  (character) names of files to be read, if NULL all files fitting 'fileFormat'
path  (character) where files should be read (folders should be written in R-style)
fileFormat  (character) may be NULL (both US and European formats will be tried), 'Eur' or 'US'
checkFormat  (logical) if TRUE: check header, remove empty columns, 1st line if all empty, set output format for each file to matrix, if rownames are increasing integeres try to use 2nd of 'columns' as rownames
returnArray  (logical) allows switching from array to list-output
columns  (NULL or character) column-headers to be extracted (if specified), 2nd value may be column with rownames (if rownames are encountered as increasing rownames)
excludeFiles  (character) names of files to exclude (only used when reading all files of given directory)
readVarColumns

Description

Reading the content of files where the number of separators (e.g., tabulation) is variable poses problems with traditional methods for reading files, like `read.table`. This function reads each line independently and then parses all separators therein. The first line is assumed to be column-headers. Finally, all data will be returned in a matrix adapted to the line with most separators and if the number of column-headers is insufficient, new (unique) column-headers will be generated. Thus, the lines may contain different numbers of elements, empty elements (i.e., tabular fields) will always get added to right of data read and their content will be as defined by argument `emptyFields` (default `NA`).

Usage

```r
readVarColumns(
  fi,
  path = NULL,
  sep = "\t",
  emptyFields = NA,
  refCo = NULL,
  supNa = NULL,
)```

Examples

```r
path1 <- system.file("extdata", package = "wrMisc")
fiNa <- c("pl01_1.csv", "pl01_2.csv", "pl02_1.csv", "pl02_2.csv")
datAll <- readCsvBatch(fiNa, path1)
str(datAll)
## batch reading of all csv files in specified path:
datAll2 <- readCsvBatch(fileNames = NULL, path = path1, silent = TRUE)
```
callFrom = NULL,
silent = FALSE
)

Arguments

fi (character) file-name
path (character) optional path
sep (character) separator (between columns)
emptyFields (NA or character) missing headers will be replaced by the content of 'empty-Fields', if NA the last column-name will be re-used and a counter added
refCo (integer) for custom choice of column to be used as row-names (default will use 1st text-column)
supNa (character) base for constructing name for columns wo names (+counter starting at 2), default column-name to left of 1st col wo colname
callFrom (character) allow easier tracking of message(s) produced
silent (logical) suppress messages

Details

Note, this functions assumes one line of header and at least one line of data ! Note, for numeric data the comma is assumed to be US-Style (as ' '). Note, that it is assumed, that any missing fields for the complete tabular view are missing on the right (ie at the end of line) !

Value

matrix (character or numeric)

See Also

for regular 'complete' data read.table

Examples

path1 <- system.file("extdata",package="wrMisc")
fiNa <- "Names1.tsv"
DatAll <- readCsvBatch(fiNa,path1)
str(DatAll)
**Description**

readXlsxBatch reads data out of multiple xlsx files, the sheet indicated by 'sheetInd' will be considered. All files must have a very similar organization of data, as this is typically the case when high-throughput measurements are automatically saved while the screen progresses. The file-names will be used to structure the data read. By default all columns with text-content may be eliminated to extract the numeric part only, which may then get organized to a 3-dim array. NOTE: requires package xlsx being installed! Uses a considerable amount of RAM! Reading multiple xlsx files does take some time.

**Usage**

```r
readXlsxBatch(
  fileNames = NULL,
  path = ".",
  fileExtension = "xlsx",
  excludeFiles = NULL,
  sheetInd = 1,
  checkFormat = TRUE,
  returnArray = TRUE,
  columns = c("Plate", "Well", "StainA"),
  simpleNames = 3,
  silent = FALSE,
  callFrom = NULL
)
```

**Arguments**

- **fileNames** (character) provide either explicit list of file-names to be read or leave NULL for reading all files ending with 'xlsx' in path specified with argument path
- **path** (character) there may be a different path for each file
- **fileExtension** (character) extension of files (default='xlsx')
- **excludeFiles** (character) names of files to exclude (only used when reading all files of given directory)
- **sheetInd** (integer) specify which sheet to extract (must be number, eg sheetInd=2 will extract always the 2nd sheet (no matter the name)
- **checkFormat** (logical) if TRUE: check header, remove empty columns, if rownames are increasing integeres it will search for first column with different entries to use as rownames
- **returnArray** (logical) allows switching from array to list-output
- **columns** (NULL or character) column-headers to be extracted (if specified, otherwise all columns will be extracted)
reduceTable

simpleNames (integer), if NULL all characters of fileNames will be maintained, otherwise allows truncating names (from beginning) to get to variable part (using .trimFromStart()), but keeping at least the number of characters indicated by this argument

silent (logical) suppress messages
callFrom (character) allows easier tracking of message(s) produced

Value

list

See Also

read.xlsx, for simple reading of xls-files under 32-bit R see also package RODBC

Examples

path1 <- system.file("extdata", package="wrMisc")
fiNa <- c("pl01_1.xlsx", "pl01_2.xlsx", "pl02_1.xlsx", "pl02_2.xlsx")
datAll <- readXlsxBatch(fiNa, path1)
str(datAll)
datAll2 <- readXlsxBatch(path = path1, silent = TRUE)
identical(datAll, datAll2)

reduceTable

Reduce table by aggregating smaller groups

Description

reduceTable treats/reduces results from table to 'nGrp' groups, optional indiv resolution of 'sepFirst' (numeric or NULL). Mainly made for reducing the number of classes for betters plots with pie

Usage

reduceTable(tab, separFirst = 4, nGrp = 15)

Arguments

tab output of table

separFirst (integer or NULL) optimal separation of n 'separFirst' groups (value < 2 or NULL will priviledge more uniform size of groups, higher values will cause small initial and larger tailing groups)
nGrp (integer) number of groups expected
regrBy1or2point

Value
numeric vector with number of counts and class-borders as names (like table).

See Also
table

Examples
set.seed(2018); dat <- sample(11:60,200,repl=TRUE)
pie(table(dat))
pie(reduceTable(table(dat),sep=NULL))
pie(reduceTable(table(dat),sep=NULL),init.angle=90,clockwise=TRUE,col=rainbow(20)[1:15],cex=0.8)

regrBy1or2point Rescaling according to reference data using linear regression.

Description
regrBy1or2point does rescaling: linear transform simple vector 'inDat' that (mean of) elements of names cited in 'refLst' will end up as values 'regrTo'. Regress single vector according to 'refLst' (describing names of inDat). If 'refLst' contains 2 groups, the 1st group will be set to the 1st value of 'regrTo' (and the 2nd group of 'refLst' to the 2nd 'regrTo')

Usage
regrBy1or2point(
inDat,
refLst,
regrTo = c(1, 0.5),
silent = FALSE,
callFrom = NULL
)

Arguments
inDat matrix or data.frame
refLst list of names existing in inDat (one group of names for each value in 'regrTo'), to be transformed in values precised in 'regrTo'; if no matches to names of 'inDat' found, the 2 lowest and/or highest highest values will be chosen
regrTo (numeric,length=2) range (at scale 0-1) of target-values for mean of elements cited in 'refLst'
silent (logical) suppress messages
callFrom (character) allows easier tracking of message(s) produced
Value

normalized matrix

See Also

adjBy2ptReg, regrMultBy1or2point

Examples

set.seed(2016); dat1 <- 1:50 +(1:50)*round(runif(50),1)
names(dat1) <- 1:length(dat1)
reg1 <- regrBy1or2point(dat1,refLst=c("2","49"))
plot(reg1,dat1)

Description

regrMultBy1or2point regresses each col of matrix according to 'refLst' (describing rownames of inDat). If 'refLst' contatins 2 groups, the 1st group will be set to the 1st value of 'regrTo' (and the 2nd group of 'refLst' to the 2nd 'regrTo')

Usage

regrMultBy1or2point(
inDat,
refLst,
regrTo = c(1, 0.5),
silent = FALSE,
callFrom = NULL
)

Arguments

inDat matrix or data.frame
refLst list of names existing in inDat (one group of names for each value in 'regrTo'), to be transformed in values precised in 'regrTo'; if no matches to names of 'inDat' found, the 2 lowest and/or highest highest values will be chosen
regrTo (numeric,length=2) range (at scale 0-1) of target-values for mean of elements cited in 'refLst'
silent (logical) suppress messages
callFrom (character) allow easier tracking of message(s) produced
renameColumns

Value

normalized matrix

See Also

adjBy2ptReg, regrBy1or2point

Examples

set.seed(2016); dat2 <- round(cbind(1:50 +(1:50)*rnorm(50), 2.2*(1:50) +rnorm(50,0,3)),1)
rownames(dat2) <- 1:nrow(dat2)
reg1 <- regrBy1or2point(dat2[,1],refLst=list(as.character(5:7),as.character(44:45)))
reg2 <- regrMultBy1or2point(dat2,refLst=list(as.character(5:7),as.character(44:45))
plot(dat2[,1],reg2[,1])
identical(reg1,reg2[,1])
identical(dat2[,1],reg2[,1])

renameColumns renames columns of 'refMatr' using 2-column matrix (or data.frame) indicating old and new names (for replacement).

Usage

renameColumns(refMatr, newName, silent = FALSE, callFrom = NULL)

Arguments

refMatr matrix (or data.frame) where column-names should be changed
newName (matrix of character) giving correspondence of old to new names (number of lines must match number of columns of 'refMatr')
silent (logical) suppres messages
callFrom (character) allows easier tracking of message(s) produced

Value

matrix (or data.frame) with renamed columns
replNAbyLow

Examples

\begin{verbatim}
ma <- matrix(1:8,ncol=4,dimnames=list(1:2,LETTERS[1:4]))
replBy1 <- cbind(new=c("dd","bb","z_"),old=c("D","B","zz"))
replBy2 <- matrix(c("D","B","zz","dd","bb","z_"),ncol=2)
replBy3 <- matrix(c("X","Y","zz","xx","yy","z_"),ncol=2)
renameColumns(ma,replBy1)
renameColumns(ma,replBy2)
renameColumns(ma,replBy3)
\end{verbatim}

ReplNAbyLow

Replace NAs by low values

Description

With several screening techniques used in high-throughput biology values at/below detection limit are returned as NA. However, the resultant NA-values may be difficult to analyse properly, simply ignoring NA-values may not be a good choice. When (technical) replicate measurements are available, one can look for cases where one gave an NA while the other did not with the aim of investigating such 'NA-neighbours'. replNAbyLow locates and replaces NA values by (random) values from same line & same group 'grp'. The origin of NAs should be predominantly absence of measure (quantitation) due to signal below limit of detection and not saturation at upper detection limit or other technical problems. Note, this approach may be not optimal if the number of NA-neighbours is very low. Replacement is done -depending on argument 'unif'- by Gaussian random model based on neighbour values (within same group), using their means and sd, or a uniform random model (min and max of neighbour values). Then numeric matrix (same dim as 'x') with NA replaced is returned.

Usage

replNAbyLow(
x, grp, quant = 0.8, signific = 3, unif = TRUE, absOnly = FALSE, seed = NULL, silent = FALSE, callFrom = NULL)

Arguments

x (numeric matrix or data.frame) main input
grp (factor) to organize replicate columns of (x)
quant (numeric) quantile form 'neighbour' values to use as upper limit for random values
replPlateCV

Description

replPlateCV gets CVs of replicates from list of 2 or 3-dim arrays (where 2nd dim is replicates, 3rd dim may be channel). Note: all list-elements of must MUST have SAME dimensions! When treating data from microtiter plates (eg 8x12) data are typically spread over multiple plates, ie initial matrixes that are the organized into arrays. Returns matrix or array (1st dim is intraplate-position, 2nd .. plate-group/type, 3rd .. channels)

Usage

replPlateCV(lst, callFrom = NULL)

Arguments

lst list of matrixes : suppose lines are independent elements, columns are replicates of the 1st column. All matrixes must have same dimensions

callFrom (character) allows easier tracking of messages produced

Value

matrix or array (1st dim is intraplate-position, 2nd .. plate-group/type, 3rd .. channels)
rmDupl2colMatr

Remove lines of matrix redundant /duplicated for 1st and 2nd column

Description

rmDupl2colMatr removes lines of matrix that are redundant /duplicated for 1st and 2nd column (irrespective of content of their columns). The first occurrence of redundant /duplicated elements is kept.

Usage

rmDupl2colMatr(mat, useCol = c(1, 2))

Arguments

mat (matrix or data.frame) main input
useCol (integer, length=2) columns to consider/use when looking for duplicated entries

Value

matrix with duplicated lines removed

See Also

unlist

Examples

mat <- matrix(1:12, ncol=3)
mat[3,1:2] <- mat[1,1:2]
rmDupl2colMatr(mat)
**Description**

`rowCVs` returns CV for values in each row (using speed optimized standard deviation). Note: NaN values get replaced by NA.

**Usage**

`rowCVs(dat, autoconvert = NULL)`

**Arguments**

- `dat` (numeric) matrix
- `autoconvert` (NULL or character) allows converting simple vectors in matrix of 1 row (autoconvert="row")

**Value**

(numeric) vector with CVs for each row of `dat`

**See Also**

`colSums, rowGrpCV, rowSds`

**Examples**

```r
set.seed(2016); dat1 <- matrix(c(runif(200)+rep(1:10,20)),ncol=10)
head(rowCVs(dat1))
```

---

**Description**

`rowGrpCV` calculates CVs for matrix with multiple groups of data, ie one CV for each group of data. Groups are specified as columns of `x` in `grp` (so length of `grp` should match number of columns of `x`, NAs are allowed)

**Usage**

`rowGrpCV(x, grp, means = NULL, listOutp = FALSE)`
rowGrpMeans

Arguments

- **x**: numeric matrix where replicates are organized into separate columns
- **grp**: (factor) defining which columns should be grouped (considered as replicates)
- **means**: (numeric) alternative values instead of means by .rowGrpMeans()
- **listOutp**: (logical) if TRUE, provide output as list with $CV$, $mean$ and $n$

Value

matrix of CV values

See Also

rowCVs, arrayCV, replPlateCV

Examples

set.seed(2016); dat1 <- matrix(c(runif(200)+rep(1:10,20)),ncol=10)
head(rowGrpCV(dat1,gr=gl(4,3,labels=LETTERS[1:4])[2:11]))

Description

rowGrpMeans calculates column-means for matrix with multiple groups of data, ie similar to rowMeans but one mean for each group of data. Groups are specified as columns of `x` in `grp` (so length of `grp` should match number of columns of `x`, NAs are allowed).

Usage

rowGrpMeans(x, grp)

Arguments

- **x**: matrix or data.frame
- **grp**: (character or factor) defining which columns should be grouped (considered as replicates)

Value

matrix with mean values

See Also

rowSds, colSums
Examples

```r
set.seed(2016); dat1 <- matrix(c(runif(200)+rep(1:10,20)),ncol=10)
head(rowGrpMeans(dat1,gr=gl(4,3,labels=LETTERS[1:4])[2:11]))
```

---

**rowGrpSds**

*Per line and per group sd-values*

**Description**

`rowGrpSds` calculate Sd (standard-deviation) for matrix with multiple groups of data, ie one sd for each group of data. Groups are specified as columns of `x` in `grp` (so length of `grp` should match number of columns of `x`, NAs are allowed).

**Usage**

```r
rowGrpSds(x, grp)
```

**Arguments**

- `x` 
  matrix where replicates are organized into separate columns
- `grp` 
  (character or factor) defining which columns should be grouped (considered as replicates)

**Value**

matrix of sd values

**See Also**

`rowGrpMeans`, `rowCVs`, `rowSEMs`, `sd`

**Examples**

```r
set.seed(2016); dat1 <- matrix(c(runif(200)+rep(1:10,20)),ncol=10)
head(rowGrpSds(dat1,gr=gl(4,3,labels=LETTERS[1:4])[2:11]))
```
Description

rowMedSds determines the standard error (sd) of the median for each row by bootstrapping each row of 'dat'. Note: requires package boot.

Usage

rowMedSds(dat, nBoot = 99)

Arguments

dat (numeric) matrix, main input

nBoot (integer) number of iterations for bootstrap

Value

(numeric) vector with estimated standard errors

See Also

boot

Examples

set.seed(2016); dat1 <- matrix(c(runif(200)+rep(1:10,20)),ncol=10)
rowMedSds(dat1) ; plot(rowSds(dat1),rowMedSds(dat1))

Description

rowSds is speed optimized sd (takes matrix or data.frame and treats each line as set of data for sd equiv to apply(dat,1,sd). NAs are ignored from data unless entire line NA). Speed improvements may be seen at more than 100 lines. Note: NaN instances will be transformed to NA.

Usage

rowSds(dat)

Arguments

dat matrix (or data.frame) with numeric values (may contain NAs)


**Value**

numeric vector of sd values

**See Also**

sd

**Examples**

```r
set.seed(2016); dat1 <- matrix(c(runif(200)+rep(1:10,20)),ncol=10)
rowSds(dat1)
```

---

**Description**

`rowSEMs` speed optimized SEM (standard error of the mean) for each row. The function takes a matrix or data.frame and treats each row as set of data for SEM; NAs are ignored from data. Note: NaN instances will be transformed to NA

**Usage**

```
rowSEMs(dat)
```

**Arguments**

- `dat` matrix or data.frame

**Value**

numeric vector with SEM values

**See Also**

`rowSds`, `colSds`, `colSums`

**Examples**

```r
set.seed(2016); dat1 <- matrix(c(runif(200)+rep(1:10,20)),ncol=10)
head(rowSEMs(dat1))
```
searchDataPairs

Search duplicated data over multiple columns, ie pairs of data

Description

searchDataPairs searches matrix for columns of similar data, i.e. 'duplicate' values in separate columns or very similar columns if 'realDupsOnly'=FALSE. Initial distance measures will be normalized either to diagonale (normRange=TRUE) of 'window' or to the real max distance observed (equal or less than diagonale). Return data.frame with names for sample-pair, percent of identical values (100 for complete identical pair) and relative (Euclidean) distance (ie max dist observed =1.0). Note, that low distance values do not necessarily imply correlating data.

Usage

searchDataPairs(
  dat,
  disThr = 0.01,
  byColumn = TRUE,
  normRange = TRUE,
  altNa = NULL,
  realDupsOnly = TRUE,
  silent = FALSE,
  callFrom = NULL
)

Arguments

dat      matrix or data.frame

disThr   (numeric) threshold to decide when to report similar data (applied on normalized distances, low val fewer reported), applied on normalized distances (norm to diagonale of all data for best relative 'unbiased' view)

byColumn (logical) rotates main input by 90 degrees (using t), thus allows to read by rows instead of by columns

normRange (logical) normize each columns separately if TRUE

altNa    (character, default NULL) vector with alternative names (for display)

realDupsOnly (logical) if TRUE will consider equal values only, otherwise will also consider very close values (based on argument disThr)

silent   (logical) suppres messages

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

Value

data.frame with names for sample-pair, percent of identical values (100 for complete identical pair) and rel (Euclidean) distance (ie max dist observed =1.0)
searchLinesAtGivenSlope

Search points forming lines at given slope

Description

searchLinesAtGivenSlope searches among set of points (2-dim) those forming line(s) with user-defined slope ('coeff'), ie search optimal (slope-) offset parameter(s) for (regression) line(s) with given slope ('coef'). Note: larger data-sets: segment residuals to 'coeff' & select most homogenous

Usage

searchLinesAtGivenSlope(
  dat,
  coeff = 1.5,
  filtExtr = c(0, 1),
  minMaxDistThr = NULL,
  lmCompare = TRUE,
  indexPoints = TRUE,
  displHist = FALSE,
  displScat = FALSE,
  bestCluByDistRat = TRUE,
  neighbDiLim = NULL,
  silent = FALSE,
  debugM = FALSE,
  callFrom = NULL
)

Arguments

dat matrix or data.frame
coeff (numeric) slope to consider
filtExtr (integer) lower & upper quantile values, remove points with extreme deviation to offset=0, (if single value: everything up to or after will be used)
minMaxDistThr (logical) optional minimum and maximum distance threshold
lmCompare (logical) add'l fitting of linear regression to best results, return offset AND slope based on lm fit
indexPoints (logical) return results as list with element 'index' specifying retained points
displHist (logical) display histogram of residues
displScat (logical) display (simple) scatter plot
bestCluByDistRat (logical) initial selection of decent clusters based on ratio overallDist/averNeighbDist
(or by CV & cor)
neighbDiLim (numeric) additional threshold for (trimmed mean) neighbour-distance
silent (logical) suppress messages
debugM (logical) for bug-tracking: more/enhanced messages
callFrom (character) allow easier tracking of message(s) produced

Value

matrix of line-characteristics (or if indexPoints is TRUE then list (line-characteristics & index & lm-results)

Examples

set.seed(2016); ra1 <- runif(300)
dat1 <- cbind(x=round(c(1:100+ra1[1:100]/5,4*ra1[1:50]),1),
y=round(c(1:100+ra1[101:200]/5,4*ra1[101:150]),1))
(li1 <- searchLinesAtGivenSlope(dat1,coeff=1))
**Arguments**

- **frag** (matrix) 2 columns defining begin- and end-sites (as integer values)
- **fullSize** (integer) optional max size used for figure (x-axis)
- **sortByHead** (logical) sort by begin-sites (if TRUE) or sort by end-sites
- **useTit** (character) custom title
- **useCol** (character) specify colors, if numeric vector will be considered as score values
- **displNa** (character) display names of edges (figure may get crowded)
- **useCex** (numeric) expansion factor

**Value**

matrix with mean values

**See Also**

`buildTree`, `countSameStartEnd`, `contribToContigPerFrag`.

**Examples**

```r
defrag2 <- cbind(beg=c(2,3,7,13,15,7,3,7,5,7),end=c(6,12,8,18,20,20,19,12,12,4))
simpleFragFig(defrag2, fullSize=21, sortByHead=TRUE)
buildTree(defrag2)
```

---

**singleLineAnova**

2-factorial Anova on single line of data

**Description**

singleLineAnova runs 2-factorial Anova on a single line of data (using `aov` from package `stats`) using a model with two factors (without factor-interaction) and extracts the corresponding p-value.

**Usage**

```r
singleLineAnova(dat, fac1, fac2, inclInteraction = TRUE)
```

**Arguments**

- **dat** numeric vector
- **fac1** (character or factor) vector describing grouping elements of dat for first factor, must be of same length as fac2
- **fac2** (character or factor) vector describing grouping elements of dat for second factor, must be of same length as fac1
- **inclInteraction** (logical) decide if factor-interactions (eg synergy) should be included to model
Value

(uncorrected) p for factor ’Pr(>F)’ (see aov)

See Also

aov, anova: for repeated tests including eBayes see test2factLimma

Examples

set.seed(2012); dat <- round(runif(8),1)
singleLineAnova(dat,gl(2,4),rep(1:2,4))

sortBy2CategorAnd1IntCol

Sort matrix by two categorical and one integer columns

Description

sortBy2CategorAnd1IntCol sorts matrix ’mat’ subsequently by categorical and numerical columns of ’mat’, ie lines with identical values for categor are sorted by numeric value.

Usage

sortBy2CategorAnd1IntCol(  
  mat,  
  categCol,  
  numCol,  
  findNeighb = TRUE,  
  decreasing = FALSE,  
  silent = FALSE,  
  callFrom = NULL  
)

Arguments

mat matrix (or data.frame) from which by 2 columns will be selected for sorting
categCol (integer or character) which columns of ’mat’ to be used as categorical columns
numCol (integer or character) which column of ’mat’ to be used as integer columns
findNeighb (logical) if ’findNeighb’ neighbour cols according to ’numCol’ will be identified as groups & marked in new col ’neiGr’, orphans marked as NA
decreasing (logical) order of sort
silent (logical) suppress messages
callFrom (character) allow easier tracking of message(s) produced
stableMode

Value

sorted matrix (same dimensions as 'mat')

Examples

mat <- cbind(aa=letters[c(3,rep(7:8,3:4),4,4:6,7)],bb=LETTERS[rep(1:5,c(1,3,4,4,1))],
  nu=c(23:21,23,21,22,18:12))
mat[c(3:5,1:2,6:9,13:10),]
sortBy2CategorAnd1IntCol(mat,cate=c("bb","aa"),num="nu",findN=FALSE,decr=TRUE)
sortBy2CategorAnd1IntCol(mat,cate=c("bb","aa"),num="nu",findN=TRUE,decr=FALSE)

stableMode

Estimate mode (most frequent value)

Description

Estimate mode (most frequent value); rounding to range of numbers used as number of significant values ('rangeSign') to better estimate continuous data. This function uses the package BBmisc.

Usage

stableMode(
  x,
  histLike = FALSE,
  rangeSign = 1:6,
  nCl = 50,
  callFrom = NULL,
  silent = FALSE
)

Arguments

x  numeric vector
histLike  (logical) if TRUE, will search context dependent, ie like most frequent class of histogram. Using this mode the search will be refined if either 80 percent of values in single class or >50 percent in single
rangeSign  (integer) range of numbers used as number of significant values
nCl  (integer) defines the number of classes used (if 'histLike'=TRUE), very 'critical' parameter, may change results in strong way !! Note: higher values for 'nCl' will finally loose advantage of histLike-type search of mode !
callFrom  (character) allow easier tracking of message(s) produced
silent  (logical) suppress messages

Value

mode value (numeric)
See Also

`cut`, `hist`

Examples

```r
set.seed(2012); dat <- round(c(rnorm(50), runif(100)), 3)
stableMode(dat)
```

```r
---
stdErrMedBoot  Standard error of median by boot-strap
---

Description

`stdErrMedBoot` estimate standard error of median by boot-strap approach. Note: requires package `boot`

Usage

`stdErrMedBoot(x, nBoot = 99)`

Arguments

- `x` (numeric) vector to estimate median and it’s standard error
- `nBoot` (integer) number for iterations

Value

(numeric) vector with estimated standard error

See Also

`boot`

Examples

```r
set.seed(2014); ra1 <- c(rnorm(9, 2, 1), runif(8, 1, 2))
rat1 <- ratioAllComb(ra1[1:9], ra1[10:17])
median(rat1); stdErrMedBoot(rat1)
```
summaryCols

Summarize columns (as median, mean, min, last or other methods)

Description

summaryCols summarizes all columns of matrix (or data.frame). In case of text-columns the sorted middle (~median) will be given, unless ‘maxLast’, ‘minLast’, ‘maxLast’, ‘maxAbsLast’ or ‘minLast’. Consider only last column of ‘matr’: choose from all columns the line where (max of) last col is at min; ‘medianComplete’ or ‘meanComplete’ considers only lines/rows where no NA occur (NA have influence other columns !)

Usage

summaryCols(matr, meth = "median", silent = FALSE, callFrom = NULL)

Arguments

matr data.frame matrix of data to be summarized by column (may do different method for text and numeric columns)

meth (character) summarization method (eg ‘maxLast’, ‘minLast’, ‘maxLast’, ‘maxAbsLast’, ‘minLast’, ‘medianComplete’ or ‘meanComplete’)

silent (logical) suppress messages

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

Value

vector with summary for each column

See Also

rowMeans in colSums

Examples

t1 <- matrix(round(runif(30,1,9),nc=3)); rownames(t1) <- letters[c(1:5,3:4,6:4)]
summaryCols(t1,me="median")
t(sapply(by(t1,rownames(t1), function(x) x), summaryCols,me="maxLast"))
t3 <- data.frame(ref=rep(11:15,3),tx=letters[1:15],
    matrix(round(runif(30,-3,2),1),ncol=2),stringsAsFactors=FALSE)
by(t3,t3[,1],function(x) x)
t(sapply(by(t3,t3[,1],function(x) x), summaryCols,me="maxAbsLast"))
tableToPlot

Print matrix-content as plot

Description

tableToPlot prints all columns of matrix in plotting region for easier inclusion to reports (default values are set to work for output as A4-sized pdf). This function was made for integrating listings of text to graphical output to devices like png, jpeg or pdf. This function was initially designed for listings with small/medium 1st col (eg counter or index), 2nd & 3rd col small and long 3rd col (like file paths). Obviously, the final number of lines one can pack and still read correctly into the graphical output depends on the size of the device (on a pdf of size A4 one can pack up to apr. 110 lines). Of course, Sweave, combined with LaTeX, provides a powerful alternative for wrapping text to pdf-output (and further combining text and graphics). Note: The final result on pdf devices may vary depending on screen-size (ie with of current device), the parameters 'colPos' and 'titOffS' may need some refinements. Note: In view of typical page/figure layouts like A4, the plotting region will be split to avoid too wide spacing between rows with less than 30 rows.

Usage

tableToPlot(
    matr, 
    colPos = c(0.05, 0.35, 0.41, 0.56), 
    useCex = 0.7, 
    useAdj = c(0, 1, 1, 0), 
    titOffS = 0, 
    useCol = 1, 
    silent = FALSE, 
    callFrom = NULL
)

Arguments

matr (matrix) main (character) matrix to display

colPos (numeric) position of columns on x-scale (from 0 to 1)

useCex (numeric) cex expansion factor for size of text (may be different for each column)

useAdj (numeric) left/center/right alignment for text (may be different for each column)

titOffS (numeric) offset for title line (relative to 'colPos')

useCol color specification for text (may be different for each column)

silent (logical) suppress messages

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

Value

NULL (no R-object returned), print 'plot' in current device only
See Also

Sweave for more flexible framework

Examples

```r
## as example let's make a listing of file-names and associated parameters in current directory
mat <- dir()
mat <- cbind(no=1:length(mat),fileName=mat,mode=file.mode(mat),si=round(file.size(mat)/1024),path=getwd())
## Now, we wrap all text into a figure (which could be saved as jpg etc)
tableToPlot(mat[,,-1],colPos=c(0.01,0.4,0.46,0.6),titOffS=c(0.05,-0.03,-0.01,0.06))
tableToPlot(mat,colPos=c(0,0.16,0.36,0.42,0.75),useAdj=0.5,titOffS=c(-0.01,0,-0.01,0,-0.1))
```

test2factLimma  2-factorial limma-style t-test

Description

test2factLimma runs 2 fact test on data 'datMatr' which should already be organized as matrix (rows = genes) using eBayes Note: this function uses the Bioconductor package limma

Usage

```r
test2factLimma(
  datMatr,
  fac1,
  fac2,
  testSynerg = TRUE,
  testOrientation = "=",
  addResults = c("lfdr", "FDR", "Mval", "means"),
  addGenes = NULL,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datMatr</td>
<td>matrix or data.frame with lines as independent series of measures (eg different genes)</td>
</tr>
<tr>
<td>fac1</td>
<td>(character or factor) vector describing grouping elements of each line of 'datMatr' for first factor, must be of same length as fac2</td>
</tr>
<tr>
<td>fac2</td>
<td>(character or factor) vector describing grouping elements of each line of 'datMatr' for second factor, must be of same length as fac1</td>
</tr>
<tr>
<td>testSynerg</td>
<td>(logical) decide if factor-interactions (eg synergy) should be included to model</td>
</tr>
<tr>
<td>testOrientation</td>
<td>(character) default (or any non-recognized input) '='; otherwise either '&gt;'; 'greater'; 'sup'; 'upper' or '&lt;'; 'inf'; 'lower'</td>
</tr>
</tbody>
</table>
transpGraySca

Make single vector gray-gradient

Description

This function helps making gray-gradients. Note: The resulting color gradient does not seem linear to the human eye, you may try gray.colors instead

Usage

transpGraySca(startGray = 0.2, endGrey = 0.8, nSteps = 5, transp = 0.3)

Arguments

startGray (numeric) gray shade at start
endGrey (numeric) gray shade at end
nSteps (integer) number of levels
transp (numeric) transparency alpha

debugResults (character) vector defining which types of information should be included to output, may be 'Ifdr', 'FDR' (for BY correction), 'Mval' (M values), 'means' (matrix with mean values for each group of replicates)
addGenes (matrix or data.frame) additional information to add to output
silent (logical) suppress messages
callFrom (character) allow easier tracking of message(s) produced

Value

object of class "MAArrayLM" (from limma)

See Also

single line testing eBayes, for single line testing without eBayes see test2factLimma

Examples

## example for testing change of ratio for 4 sets (AA-DD) of pairs of data
set.seed(2017); t8 <- matrix(round(rnorm(160,10,0.4),2),ncol=8,
  dimnames=list(letters[1:20],c("AA1","BB1","CC1","DD1","AA2","BB2","CC2","DD2")))
t8[3:6,1:2] <- t8[3:6,1:2]+3 # augment lines 3:6 (c-f) for AA1&BB1
t8[5:8,5:6] <- t8[5:8,5:6]+3 # augment lines 5:8 (e-h) for AA2&BB2 (c,d,g,h should be found)
## via MAobj
maOb8 <- makeMAList(t8,MAf=gl(2,4,labels=c("R","G"))
fit8b <- test2factLimma(maOb8,c(1,1,2,2),c(0,0,1,1),testS=FALSE) # same result as fit8e
limma::topTable(fit8b,coef=1,n=5) # effect for c,d,g&h
## explicit (long) way via limma:
fit8 <- limma::lmFit(maOb8, design= model.matrix(~ 0+factor(c(1,1,2,2))))
fit8e <- limma::eBayes(fit8)
limma::topTable(fit8e,coef=1,n=5) # effect for c,d,g&h

Value

character vector (of same length as x) with color encoding

See Also

gray.colors

Examples

```
layout(1:2)
col1 <- wrMisc::transpGraySca(0.8,0.3,7,0.9)
pie(rep(1,length(col1)),col=col1,main="from transpGraySca")
col2 <- gray.colors(7,0.9,0.3,alph=0.9)
pie(rep(1,length(col2)),col=col2,main="from gray.colors")
```

---

treatTxtDuplicates  Locate duplicates in text and make non-redundant

Description

treatTxtDuplicates locates duplicates in character-vector 'x' and return list (length=3) : with $init (initial), $nRed .. non-redundant text by adding number at end or beginning, and $nrLst .. list-version with indexes per unique entry. Note : NAs (if multiple) will be renamed to NA_1, NA_2

Usage

```
treatTxtDuplicates(
  x,
  atEnd = TRUE,
  sep = "_",
  onlyCorrectToUnique = FALSE,
  silent = FALSE,
  callFrom = NULL
)
```

Arguments

- **x**: (character) vector with character-entries to identify (and remove) duplicates
- **atEnd**: (logical) decide location of placing the counter (at end or at beginning of ID) (see `correctToUnique`)
- **sep**: (character) separator to add before counter when making non-redundant version
- **onlyCorrectToUnique**: (logical) if TRUE, return only vector of non-redundant
- **silent**: (logical) suppress messages
- **callFrom**: (character) allow easier tracking of message(s) produced
**triCoord**

**Value**

list with $init, $nRed, $nrLst

**See Also**

For simple correction use `correctToUnique`

**Examples**

```r
treatTxtDuplicates(c("li0",NA,rep(c("li2","li3"),2)))
correctToUnique(c("li0",NA,rep(c("li2","li3"),2)))
```

---

**triCoord**

*Pairwise x,y combinations*

**Description**

`triCoord` gets pairwise combinations for 'n' elements; returns matrix with x & y coordinates to form all pairwise groups for 1:n elements

**Usage**

`triCoord(n, side = "upper")`

**Arguments**

- `n` (integer) number of elements for making all pair-wise combinations
- `side` (character) "upper" or "lower"

**Value**

2-column matrix with indexes for all pairwise combinations of 1:n

**See Also**

`lower.tri` or `upper.tri`, simpler version `upperMaCoord`

**Examples**

```r
triCoord(4)
```
tTestAllVal  t.test on all individual values against all other values

Description

Run t.test on each indiv value of x against all its neighbours (=remaining values of same vector) in order to test if tis value is likely to belong to vector x. This represents a repeated leave-one-out testing. Multiple choices for multiple testing correction are available.

Usage

tTestAllVal(x, alph = 0.05, alternative = "two.sided", p.adj = NULL)

Arguments

x matrix or data.frame

alph (numeric) threshold alpha (passed to t.test)

alternative (character) will be passed to t.test as argument 'alternative', may be "two.sided"

p.adj (character) multiple test correction: may be NULL (no correction), "BH", "BY", "holm", "hochberg" or "bonferroni" (but not 'fdr' since this may be confounded with local false discovery rate), see p.adjust

Value

numeric vector with p-values or FDR (depending on argument p.adj)

See Also

t.test, p.adjust

Examples

set.seed(2016); x1 <- rnorm(100)
allTests1 <- tTestAllVal(x1)
hist(allTests1,breaks="FD")
uniqCountReport

Report number of unique and redundant elements (optional figure)

Description

Make report about number of unique and redundant elements of vector 'dat'. Note : fairly slow for long vectors !

Usage

uniqCountReport(
  dat,
  frL = NULL,
  plotDispl = FALSE,
  tit = NULL,
  col = NULL,
  radius = 0.9,
  sizeTo = NULL,
  clockwise = FALSE,
  silent = FALSE,
  callFrom = NULL
)

Arguments

dat  (character or numeric vector) main input where number of unique (and redundant) should be determined
frL  (logical) optional (re-)introducing results from duplicated to shorten time of execution
plotDispl  (logical) decide if pie-type plot should be produced
tit  (character) optional title in plot
col  (character) custom colors in pie
radius  (numeric) radius passed to pie
sizeTo  (numeric or character) optional reference group for size-population relative adjusting overall surface of pie
clockwise  (logical) argument passed to pie
silent  (logical) suppress messages
callFrom  (character) allow easier tracking of message(s) produced

Value

vector with counts of n (total), nUnique (wo any repeated), nHasRepeated (first of repeated), nRedundant, optional figure
upperMaCoord

See Also
correctToUnique, unique

Examples
layout(1:2)
uniqCountReport(rep(1:7,1:7),plot=TRUE)
uniqCountReport(rep(1:3,1:3),plot=TRUE,sizeTo=rep(1:7,1:7))

upperMaCoord(n)

Arguments
n (integer) number of elements for making all pair-wise combinations

Value
2-column matrix with indexes for all pairwise combinations of 1:n

See Also
lower.tri, more evolved version triCoord

Examples
upperMaCoord(4)
withinRefRange

Description

withinRefRange checks which values of numeric vector 'x' are within range +/- 'fa' x 'ref' (ie within range of reference).

Usage

withinRefRange(x, fa, ref = NULL, absRef = TRUE, asInd = FALSE)

Arguments

- **x** (matrix or data.frame)
- **fa** (numeric) absolute or relative tolerance value (numeric, length=1), interpreted according to 'absRef' as absolute or relative to 'x'(ie fa*ref)
- **ref** (numeric) (center) reference value for comparison (numeric, length=1), if not given mean of 'x' (excluding NA or non-finite values) will be used
- **absRef** (logical) return result as absolute or relative to 'x'(ie fa*ref)
- **asInd** (logical) if TRUE return index of which values of 'x' are within range, otherwise return values if 'x' within range

Value

numeric vector (containing only the values within range of reference)

Examples

```r
## within 2.5 +/- 0.7
withinRefRange(-5:6,fa=0.7,ref=2.5)
## within 2.5 +/- (0.7*2.5)
withinRefRange(-5:6,fa=0.7,ref=2.5,absRef=FALSE)
```

writeCsv

Write (and convert) csv files

Description

This functions allows to write data into csv-files (and/or tabulated txt files) from given vector or read from file 'input' and (re-)write using specified conversions. An optional filter to select columns (column-name specified via 'filterCol') is available. The output may be simultaneaously written to multiple formats, as specified in 'expTy', tabulation characters may be converted to avoid accidentally split/shift text to multiple columns. Note: Mixing '.' and ',' as comma separators via text-columns or fused text&data may cause problems lateron, though.
Usage

documentation content...
See Also

write.csv in write.table, batch reading using this package readCsvBatch

Examples

dat1 <- data.frame(ini=letters[1:5],x1=1:5,x2=11:15,t1=c("10,10","20.20","11,11","21,21","33.33"),
t2=c("10,11","20.21","kl;kl","az,az","ze.ze"))
fiNa <- file.path(tempdir(),paste("test",1:2,".csv",sep=""))
writeCsv(dat1,filename=fiNa[1])
dir(path=tempdir(),pattern="cs")
(writeCsv(dat1,replM=rbind(bad=c(";",""),replBy="_"),expTy=c("Eur"),
    returnOut=TRUE,filename=fiNa[2])))

---

XYToDiffPpm

Express difference as ppm

Description

XYToDiffPpm transforms offset (pairwise-difference) between 'x' & 'y' to ppm (as normalized difference ppm, parts per million, ie (x-y)/y ). This type of expression of differences is used eg in mass-spectrometry.

Usage

XYToDiffPpm(x, y, nSign = NULL, silent = FALSE, callFrom = NULL)

Arguments

x (numeric) typically for measured variable
y (numeric) typically for theoretical/expected value (vector must be of same length as 'x')
nSign (integer) number of significant digits in output
silent (logical) suppres messages
callFrom (character) allow easier tracking of message(s) produced

Value

numeric vector of (ratio-) ppm values

See Also

ratioToPpm for classical ppm

Examples

set.seed(2017); aa <- runif(10,50,900)
cbind(x=aa,y=aa+1e-3,ppm=XYToDiffPpm(aa,aa+1e-3,nSign=4))
Index

+Topic character
  pasteC, 89

addBeforeFileExtension, 5
adjBy2ptReg, 5, 85, 101, 102
anova, 115
aov, 114, 115
arrayCV, 6, 105, 107
asSepList, 7, 88

boot, 20, 109, 117
buildTree, 8, 27, 28, 34, 114

cbind, 10, 65, 67
cbindNR, 9
checkAvSd, 10
checkGrpOrder, 11
checkGrpOrderSEM, 12, 12
checkSimValueInSer, 13, 18, 38, 51, 53, 69
checkStrictOrder, 14
checkVectLength, 15
cleanReplicates, 15
closeMatchMatrix, 17, 53
coinPermTest, 18
colMedSds, 19
colorAccording2, 20
colSds, 21, 110
colSums, 106, 107, 110, 118
combineByEitherFactor, 22
combineOverlapInfo, 23
combineRedBasedOnCol, 24, 83, 84
combineRep1FromListToMatr, 25
combineSingleT, 26
completeArrLst, 26
contribToContigPerFrag, 9, 27, 34, 114
conv01toColNa, 28
convColorToTransp, 29
convMatr2df, 30
convToNum, 31, 71
correctToUnique, 32, 49, 50, 62, 83, 84, 122, 123, 126
countCloseToLimits, 33, 51
countSameStartEnd, 9, 34, 114
cut, 21, 36, 37, 117
cutArrayInCluLike, 35
cutAtMultiSites, 36, 79, 80
cutToNgrp, 36
diff, 38
diffCombin, 37
diffPPM, 38
dist, 75, 87, 112
duplicated, 49, 50, 55, 56, 82, 112
eBayes, 76, 78, 115, 120, 121
eImCloseCoord, 39
equalLenNumber, 40
exclExtrValues, 40
exponNormalize, 42, 85
extr1chan, 25, 27, 43
extractLast2numericParts, 44
extrColsDeX, 45, 49
extrNumericFromMatr, 46
extrSpText, 46

fdrtool, 91
filt3dimArr, 47
filterList, 45, 48
filtSizeUniq, 33, 49
findCloseMatch, 17, 18, 34, 39, 50, 52, 53
findRepeated, 25, 52, 62, 84
findSimilFrom2sets, 52
findUsableGroupRange, 54
firstLineOfDat, 55, 84
firstOfRepeated, 39, 55, 55, 56, 57, 61, 82, 84
firstOfRepLines, 10, 25, 41, 56, 56, 84
fisher.test, 92
fuseAnnotMatr, 57
<table>
<thead>
<tr>
<th>Function</th>
<th>Page/Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>fuseCommonListElem</td>
<td>58</td>
</tr>
<tr>
<td>fusePairs</td>
<td>59</td>
</tr>
<tr>
<td>get1stOfRepeatedByCol</td>
<td>41, 52, 60, 62, 84</td>
</tr>
<tr>
<td>getValuesByUnique</td>
<td>61</td>
</tr>
<tr>
<td>gray.colors</td>
<td>121, 122</td>
</tr>
<tr>
<td>grep</td>
<td>45, 64</td>
</tr>
<tr>
<td>hist</td>
<td>117</td>
</tr>
<tr>
<td>htmlSpecCharConv</td>
<td>62</td>
</tr>
<tr>
<td>justvsn</td>
<td>84, 85</td>
</tr>
<tr>
<td>lmRegrParamAndPVal</td>
<td>63</td>
</tr>
<tr>
<td>listBatchReplace</td>
<td>64</td>
</tr>
<tr>
<td>listGroupsByNames</td>
<td>64</td>
</tr>
<tr>
<td>lm, 63, 66, 92</td>
<td>84, 85</td>
</tr>
<tr>
<td>lmFit</td>
<td>76</td>
</tr>
<tr>
<td>lmSelClu</td>
<td>65</td>
</tr>
<tr>
<td>LocationTests</td>
<td>19</td>
</tr>
<tr>
<td>lower.tri</td>
<td>123, 126</td>
</tr>
<tr>
<td>lrbind</td>
<td>66</td>
</tr>
<tr>
<td>makeMAList</td>
<td>67</td>
</tr>
<tr>
<td>makeNRedMatr</td>
<td>68</td>
</tr>
<tr>
<td>match</td>
<td>69, 70</td>
</tr>
<tr>
<td>matchNamesWithReverseParts</td>
<td>69</td>
</tr>
<tr>
<td>matr2list</td>
<td>70</td>
</tr>
<tr>
<td>merge</td>
<td>58, 71–74</td>
</tr>
<tr>
<td>mergeSelCol</td>
<td>71, 73</td>
</tr>
<tr>
<td>mergeSelCol2</td>
<td>72, 72</td>
</tr>
<tr>
<td>mergeW2</td>
<td>73</td>
</tr>
<tr>
<td>minDiff</td>
<td>75</td>
</tr>
<tr>
<td>moderTest2grp</td>
<td>76, 78</td>
</tr>
<tr>
<td>moderTestXgrp</td>
<td>77</td>
</tr>
<tr>
<td>na.fail</td>
<td>79, 104</td>
</tr>
<tr>
<td>naOmit</td>
<td>79, 104</td>
</tr>
<tr>
<td>nchar</td>
<td>81</td>
</tr>
<tr>
<td>nFragments</td>
<td>36, 79</td>
</tr>
<tr>
<td>nFragments0</td>
<td>36, 80</td>
</tr>
<tr>
<td>nNonNumChar</td>
<td>80</td>
</tr>
<tr>
<td>Node</td>
<td>9</td>
</tr>
<tr>
<td>nonAmbiguousMat</td>
<td>81</td>
</tr>
<tr>
<td>nonAmbiguousNum</td>
<td>10, 56, 57, 61, 82</td>
</tr>
<tr>
<td>nonredDataFrame</td>
<td>83</td>
</tr>
<tr>
<td>nonRedundLines</td>
<td>84</td>
</tr>
<tr>
<td>normalizeThis</td>
<td>6, 43, 84</td>
</tr>
<tr>
<td>normalizeWithinArrays</td>
<td>68</td>
</tr>
<tr>
<td>numeric</td>
<td>32</td>
</tr>
<tr>
<td>order</td>
<td>14</td>
</tr>
<tr>
<td>organizeAsListOfRepl</td>
<td>25, 27, 44, 86</td>
</tr>
<tr>
<td>p.adjust</td>
<td>91, 124</td>
</tr>
<tr>
<td>par</td>
<td>29</td>
</tr>
<tr>
<td>partialDist</td>
<td>87</td>
</tr>
<tr>
<td>partUnlist</td>
<td>7, 8, 88</td>
</tr>
<tr>
<td>paste</td>
<td>89</td>
</tr>
<tr>
<td>pasteC</td>
<td>89</td>
</tr>
<tr>
<td>pie</td>
<td>99</td>
</tr>
<tr>
<td>presenceFilt</td>
<td>89</td>
</tr>
<tr>
<td>pVal2lfdr</td>
<td>90</td>
</tr>
<tr>
<td>randIndex</td>
<td>92</td>
</tr>
<tr>
<td>randIndFx</td>
<td>91</td>
</tr>
<tr>
<td>rankToContigTab</td>
<td>92</td>
</tr>
<tr>
<td>ratioAllComb</td>
<td>93</td>
</tr>
<tr>
<td>ratioToPpm</td>
<td>94, 129</td>
</tr>
<tr>
<td>read.table</td>
<td>45, 96, 97</td>
</tr>
<tr>
<td>read.xlsx</td>
<td>99</td>
</tr>
<tr>
<td>readCsvBatch</td>
<td>95, 129</td>
</tr>
<tr>
<td>readVarColumns</td>
<td>96</td>
</tr>
<tr>
<td>readXlsxBatch</td>
<td>96, 98</td>
</tr>
<tr>
<td>reduceTable</td>
<td>99</td>
</tr>
<tr>
<td>regrBy1lor2point</td>
<td>100, 102</td>
</tr>
<tr>
<td>regrMultBy1lor2point</td>
<td>101, 101</td>
</tr>
<tr>
<td>renameColumns</td>
<td>102</td>
</tr>
<tr>
<td>replNAbyLow</td>
<td>103</td>
</tr>
<tr>
<td>replPlateCV</td>
<td>7, 104, 107</td>
</tr>
<tr>
<td>rgb</td>
<td>29</td>
</tr>
<tr>
<td>rmDup12colMatr</td>
<td>105</td>
</tr>
<tr>
<td>rowCVs</td>
<td>7, 105, 106, 107, 108</td>
</tr>
<tr>
<td>rowGrpCV</td>
<td>7, 106, 106</td>
</tr>
<tr>
<td>rowGrpMeans</td>
<td>11, 107, 108</td>
</tr>
<tr>
<td>rowGrpSds</td>
<td>108</td>
</tr>
<tr>
<td>rowMedSds</td>
<td>109</td>
</tr>
<tr>
<td>rowSds</td>
<td>106, 107, 109, 110</td>
</tr>
<tr>
<td>rowSEMs</td>
<td>108, 110</td>
</tr>
<tr>
<td>sd</td>
<td>21, 108, 110</td>
</tr>
<tr>
<td>searchDataPairs</td>
<td>111</td>
</tr>
<tr>
<td>searchLinesAtGivenSlope</td>
<td>112</td>
</tr>
<tr>
<td>simpleFragFig</td>
<td>9, 34, 113</td>
</tr>
<tr>
<td>singleLineAnova</td>
<td>114</td>
</tr>
<tr>
<td>sortBy2CategorAnd1IntCol</td>
<td>115</td>
</tr>
<tr>
<td>sprintf</td>
<td>40</td>
</tr>
<tr>
<td>stableMode</td>
<td>116</td>
</tr>
<tr>
<td>stdErrMedBoot</td>
<td>117</td>
</tr>
<tr>
<td>strsplit</td>
<td>36</td>
</tr>
</tbody>
</table>
summarizeCols, \textit{69, 118}  
Sweave, \textit{119, 120}  

\textit{t, 111}  
\textit{t.test, 124}  
\textit{table, 99, 100}  
\textit{tableToPlot, 119}  
\textit{test2factLimma, 115, 120, 121}  
\textit{transpGraySca, 121}  
\textit{treatTxtDuplicates, 33, 62, 122}  
\textit{triCoord, 123, 126}  
\textit{tTestAllVal, 124}  

\textit{uniqCountReport, 125}  
\textit{unique, 33, 49, 50, 55, 57, 62, 83, 126}  
\textit{unlist, 8, 59, 88, 105}  
\textit{upperMaCoord, 123, 126}  

\textit{withinRefRange, 14, 127}  
\textit{write.table, 129}  
\textit{writeCsv, 96, 127}  

\textit{XYToDiffPpm, 94, 129}