Package ‘ribiosUtils’

March 6, 2020

Type Package

Title Utilities from and Interface to the Bioinfo-C (BIOS) Library

Version 1.5-6

Date 2020-02-27

Description Provides interface to the Bioinfo-C (internal name: BIOS) library and utilities. 'ribiosUtils' is a swiss-knife for computational biology in drug discovery, providing functions and utilities with minimal external dependency and maximal efficiency.

Depends R (>= 3.4.0), methods, grDevices, stats, utils

Suggests testthat

License GPL-3

URL https://github.com/bedapub/ribiosUtils

BugReports https://github.com/bedapub/ribiosUtils/issues

RoxygenNote 7.0.2

NeedsCompilation yes

Author Jitao David Zhang [aut, cre, ctb]
(https://orcid.org/0000-0002-3085-0909),
Clemens Broger [aut, ctb],
F.Hoffmann-La Roche AG [cph],
Junio C Hamano [cph],
Jean Thierry-Mieg [cph],
Richard Durbin [cph]

Maintainer Jitao David Zhang <jitao_david.zhang@roche.com>

Repository CRAN

Date/Publication 2020-03-06 09:40:02 UTC

R topics documented:

  allIdentical ......................................................... 4
  asNumMatrix .......................................................... 5
  assertColumnName ..................................................... 5
assertContrast ......................................................... 6
assertDesign ........................................................... 7
assertDesignContrast ................................................... 7
bedaInfo .............................................................. 8
biomicsPstorePath2URL .................................................. 9
checkFile ............................................................... 9
closedLoggerConnections ............................................... 10
columnOverlapCoefficient ............................................. 11
compTwoVecs ............................................................. 12
countTokens ............................................................. 13
createDir ............................................................... 14
cumJaccardIndex ........................................................ 15
cumOverlapCoefficient .................................................. 16
cumsumprop .............................................................. 17
cutInterval ............................................................... 17
dFFactor ................................................................. 19
dFFactor2Str ............................................................. 20
eXname ................................................................. 20
fixWidthStr ............................................................. 21
getDefaultFontFamily ................................................... 22
haltifnot ................................................................. 23
headhead ................................................................. 24
headtail ................................................................. 25
identicalMatrix .......................................................... 26
identicalMatrixValue .................................................... 26
imatch ................................................................. 27
isDir ................................................................. 28
isError ............................................................... 29
isRocheCompoundID ...................................................... 30
jaccardIndex ............................................................ 31
keepMaxStatRow .......................................................... 31
libordie ............................................................... 33
list2df ................................................................. 35
listOverlapCoefficient ................................................ 35
longdf2matrix ........................................................... 36
matchColumn ............................................................ 37
matchColumnName ........................................................ 39
matrix2longdf ........................................................... 40
mergeInfreqLevelsByCumsumprop ..................................... 41
midentical ............................................................ 42
mmatch ............................................................... 43
munion ............................................................... 45
na.false .............................................................. 46
naivePairwiseDist ......................................................... 46
ofactor ............................................................... 47
openFileDevice .......................................................... 48
overlapCoefficient ....................................................... 49
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>pAbsLog10Score</td>
<td>50</td>
</tr>
<tr>
<td>pairwiseJaccardIndex</td>
<td>51</td>
</tr>
<tr>
<td>pairwiseOverlapDistance</td>
<td>52</td>
</tr>
<tr>
<td>percentage</td>
<td>53</td>
</tr>
<tr>
<td>pQnormScore</td>
<td>53</td>
</tr>
<tr>
<td>print.BEDAinfo</td>
<td>54</td>
</tr>
<tr>
<td>pScore</td>
<td>55</td>
</tr>
<tr>
<td>putColsFirst</td>
<td>56</td>
</tr>
<tr>
<td>pwdecode</td>
<td>56</td>
</tr>
<tr>
<td>pencode</td>
<td>57</td>
</tr>
<tr>
<td>qmsg</td>
<td>58</td>
</tr>
<tr>
<td>qsystem</td>
<td>59</td>
</tr>
<tr>
<td>refactorNum</td>
<td>59</td>
</tr>
<tr>
<td>registerLog</td>
<td>60</td>
</tr>
<tr>
<td>relevels</td>
<td>62</td>
</tr>
<tr>
<td>relevelsByNamedVec</td>
<td>63</td>
</tr>
<tr>
<td>relevelsByNotNamedVec</td>
<td>64</td>
</tr>
<tr>
<td>reload</td>
<td>65</td>
</tr>
<tr>
<td>removeColumns</td>
<td>66</td>
</tr>
<tr>
<td>removeInvarCol</td>
<td>67</td>
</tr>
<tr>
<td>replaceColumnName</td>
<td>68</td>
</tr>
<tr>
<td>ribiosTempdir</td>
<td>68</td>
</tr>
<tr>
<td>ribiosTempfile</td>
<td>69</td>
</tr>
<tr>
<td>ribiosUtils</td>
<td>69</td>
</tr>
<tr>
<td>rmat</td>
<td>70</td>
</tr>
<tr>
<td>rocheCore</td>
<td>71</td>
</tr>
<tr>
<td>rowscale</td>
<td>72</td>
</tr>
<tr>
<td>rowscale.matrix</td>
<td>72</td>
</tr>
<tr>
<td>rsetdiff</td>
<td>73</td>
</tr>
<tr>
<td>scriptInit</td>
<td>74</td>
</tr>
<tr>
<td>setDebug</td>
<td>74</td>
</tr>
<tr>
<td>shortenStr</td>
<td>75</td>
</tr>
<tr>
<td>silencio</td>
<td>76</td>
</tr>
<tr>
<td>sortAndFilterByCumsumprop</td>
<td>77</td>
</tr>
<tr>
<td>sortByCol</td>
<td>77</td>
</tr>
<tr>
<td>sortByDimnames</td>
<td>78</td>
</tr>
<tr>
<td>strtoken</td>
<td>79</td>
</tr>
<tr>
<td>stubbornge</td>
<td>80</td>
</tr>
<tr>
<td>subsetByColumnName</td>
<td>81</td>
</tr>
<tr>
<td>summarizeRows</td>
<td>82</td>
</tr>
<tr>
<td>trim</td>
<td>83</td>
</tr>
<tr>
<td>uniqueLength</td>
<td>84</td>
</tr>
<tr>
<td>uniqueNonNA</td>
<td>85</td>
</tr>
<tr>
<td>verbose</td>
<td>85</td>
</tr>
<tr>
<td>whoami</td>
<td>86</td>
</tr>
<tr>
<td>writeLog</td>
<td>87</td>
</tr>
</tbody>
</table>

Index 89
allIdentical

Testing whether several objects are all identical with each other

Description

Given several objects, the function tests whether all of them are identical.

Usage

allIdentical(...)

Arguments

... Objects to be tested. Can be given as a list, or simply appending names separated by commas, see example.

Value

Logical, whether all objects are the same

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

identical

Examples

test1 <- test2 <- test3 <- LETTERS[1:3]
allIdentical(test1, test2, test3)
allIdentical(list(test1, test2, test3))

num1 <- num2 <- num3 <- num4 <- sqrt(3)
allIdentical(num1, num2, num3, num4)
asNumMatrix

Convert string-valued data frame or matrix into a numeric matrix

Description

Convert string-valued data frame or matrix into a numeric matrix

Usage

asNumMatrix(x)

Arguments

x

A data.frame or matrix, most likely with string values

Value

A numeric matrix with the same dimension

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

testDf <- data.frame(a=c("2.34", "4.55"), b=c("7.33", "9.10"))
asNumMatrix(testDf)

testMatrix <- matrix(c("2.34", "4.55", "9E-3", "-2.44", "7.33", "9.10"), nrow=2)
asNumMatrix(testMatrix)

assertColumnName

Assert whether the required column names exist

Description

The function calls `matchColumnName` internally to match the column names.

Usage

assertColumnName(data.frame.cols, reqCols, ignore.case = FALSE)
### assertContrast

Check dimensionality of contrast matrix

#### Usage

```r
assertContrast(design, contrast)
```

#### Arguments

- `design`  
  - Design matrix
- `contrast`  
  - Contrast matrix

#### Value

Side effect is used: the function stops if the `ncol(design)` does not equal `nrow(contrast)`

#### Examples

```r
design <- matrix(1:20, ncol=5)
contrast <- matrix(c(-1,1,0,0,0, 0,1,0,-1,0, 0,0,1,0,-1), nrow=5)
assertContrast(design, contrast)
```
assertDesign

Check dimensionality of design matrix

Description
Check dimensionality of design matrix

Usage
assertDesign(nsample, design)

Arguments
nsample  Integer, number of samples
design   Design matrix

Value
Side effect is used: the function stops if sample size does not equal ncol(matrix)

Examples

nsample <- 4
design <- matrix(1:20, ncol=5)
assertDesign(nsample, design)

assertDesignContrast
Check dimensionality of both design and contrast matrix

Description
Check dimensionality of both design and contrast matrix

Usage
assertDesignContrast(nsample, design, contrast)

Arguments
nsample  Integer, number of samples
design   Design matrix
contrast  Contrast matrix
Value

Side effect is used: the function stops if there are errors in the dimensionalities

See Also

assertDesign, assertContrast

Examples

```r
nsample <- 4
design <- matrix(1:20, ncol=5)
contrast <- matrix(c(-1,1,0,0,0, 0,1,0,-1,0), nrow=5)
assertDesignContrast(nsample, design, contrast)
```

---

### bedaInfo

**Print BEDA project information**

#### Description

Print BEDA project information

#### Usage

```r
bedaInfo()
```

#### Value

A list, including pstore path, URL, git address, and user id. The function is used at the end of the Rmarkdown report to print relevant information to help other colleagues finding relevant resources.

#### Examples

```r
if(interactive()) {bedaInfo()}
```
biomicsPstorePath2URL  Translate BiOomics-Pathology pstore path to URL

Description

Translate BiOomics-Pathology pstore path to URL

Usage

biomicsPstorePath2URL(path)

Arguments

path  Unix path

Value

Character string of biomics pstore path The URL is only visible inside Roche

Examples

biomicsPstorePath2URL("/pstore/data/biomics/")

checkFile  Check whether file(s) exist

Description

checkFile checks whether file exists, assertFile stops the program if files do not exist

Usage

checkFile(...)

assertFile(...)

Arguments

...  Files to be checked

Details

assertFile is often used in scripts where missing a file would cause the script fail.
chosenFew

Value

checkFile returns logical vector. assertFile returns an invisible TRUE if files exist, otherwise halts and prints error messages.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

isDir and assertDir

Examples

myDesc <- system.file("DESCRIPTION", package="ribiosUtils")
myNEWS <- system.file("NEWS", package="ribiosUtils")
checkFile(myDesc, myNEWS)
assertFile(myDesc, myNEWS)

chosenFew(vec, start = 3, end = 1, collapse = ",")

Description

Print the chosen few (the first and the last) items of a long vector

Usage

Arguments

vec A vector of characters or other types that can be cast into characters
start Integer, how many elements at the start shall be printed
end Integer, how many elements at the end shall be printed
collapse Character used to separate elements

Value

A character string ready to be printed

Note

In case the vector is shorter than the sum of start and end, the whole vector is printed.
closeLoggerConnections

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

lvec1 <- 1:100
chosenFew(lvec1)
chosenFew(lvec1, start=5, end=3)

svec <- 1:8
chosenFew(svec)
chosenFew(svec, start=5, end=4)

closeLoggerConnections

Close connections to all loggers This function closes all open connections set up by loggers It is automatically run at the end of the R session (setup by registerLog)

Description

Close connections to all loggers This function closes all open connections set up by loggers It is automatically run at the end of the R session (setup by registerLog)

Usage

closeLoggerConnections()

Value

Invisible NULL. Only side effect is used.

See Also

registerLog
Pairwise jaccard/overlap coefficient can be calculated efficiently using matrix pairwise overlap coefficient of binary matrix by column.

**Description**

Pairwise jaccard/overlap coefficient can be calculated efficiently using matrix pairwise overlap coefficient of binary matrix by column.

**Usage**

```r
columnOverlapCoefficient(x, y = NULL)
```

**Arguments**

- `x`: An integer matrix, other objects will be coerced into a matrix.
- `y`: An integer matrix, other objects will be coerced into a matrix. In case of `NULL`, pairwise overlap coefficients by column of `x` is returned.

**Value**

A matrix of column-wise pairwise overlap coefficients of the binary matrix. **NaN** is reported when neither of the columns have any non-zero element.

**Examples**

```r
set.seed(1887)
testMatrix1 <- matrix(rbinom(120, 1, 0.2), nrow=15)
columnOverlapCoefficient(testMatrix1)
testMatrix2 <- matrix(rbinom(150, 1, 0.2), nrow=15)
testMatrix12Poe <- columnOverlapCoefficient(testMatrix1, testMatrix2)
```

**compTwoVecs**

**Compare two vectors by set operations**

**Description**

Basic set operations are used to compare two vectors.

**Usage**

```r
compTwoVecs(vec1, vec2)
```
**countTokens**

**Arguments**

- vec1: A vector of atomic types, e.g. integers, characters, etc.
- vec2: A vector of the same type as vec1

**Value**

A vector of six integer elements

- vec1.setdiff: Number of unique items only in vec1 but not in vec2
- intersect: Number of items in both vec1 and vec2
- vec2.setdiff: Number of unique items only in vec2 but not in vec1
- vec1.ulen: Number of unique items in vec1
- vec2.ulen: Number of unique items in vec2
- union: Number of unique items in vec1 and vec2

**Author(s)**

Jitao David Zhang <jitao_david.zhang@roche.com>

**Examples**

```r
year1 <- c("HSV", "FCB", "BVB", "S04", "FCN")
year2 <- c("HSV", "FCK", "S04")
compTwoVecs(year1, year2)
```

---

**countTokens**  
*Count tokens by splitting strings*

**Description**

Count tokens by splitting strings

**Usage**

```r
countTokens(str, split = "\t", ...)
```

**Arguments**

- str: A character string vector
- split: Character used to split the strings
- ...: Other parameters passed to the strsplit function
Value

Integer vector: count of tokens in the strings

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

`strsplit` to split strings, or a convenient wrapper `strtoken` in this package.

Examples

```r
myStrings <- c("HSV\t1887\tFavorite", "FCB\t1900", "FCK\t1948")
countTokens(myStrings)

# the function deals with factors as well
countTokens(factor(myStrings))
```

---

createDir

Create a directory if it does not exist, and then make sure the creation was successful.

Description

The function is particularly useful for scripting.

Usage

```r
createDir(dir, showWarnings = FALSE, recursive = FALSE, mode = "0777")
```

Arguments

- `dir` Directory name
- `showWarnings` Passed to `dir.create`
- `recursive` Passed to `dir.create`
- `mode` Passed to `dir.create`

Value

Directory name (invisible)

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>
cumJaccardIndex

Examples

tempdir <- tempdir()
createDir(tempdir)

cumJaccardIndex(list)
cumJaccardDistance(list)

Description

Cumulative Jaccard Index

Usage

cumJaccardIndex(list)
cumJaccardDistance(list)

Arguments

list                  A list of characters or integers

Value

The cumulative Jaccard Index, a vector of values between 0 and 1, of the same length as the input list.
The cumulative Jaccard Index is calculated by calculating the Jaccard Index of element $i$ and the union of elements between 1 and $i-1$. The cumulative Jaccard Index of the first element is set as 0.0.
The cumulative Jaccard distance is defined in almost the same way, with the only difference the distance is returned. The value of the first element is 1.0.

Note

An advantage of using cumulative overlap coefficient over cumulative Jaccard Index is that it is monotonic: the value is guaranteed to decrease from 1 to 0, whereas the cumulative Jaccard Index may not be monotonic.

See Also

cumOverlapCoefficient
**Examples**

```r
cumJaccardIndex(myList)
cumJaccardDistance(myList)
```

---

**cumOverlapCoefficient**  
_Cumulative overlap coefficient_

**Description**

Cumulative overlap coefficient

**Usage**

```r
cumOverlapCoefficient(list)  
cumOverlapDistance(list)
```

**Arguments**

- `list`  
  A list of characters or integers

**Value**

The cumulative overlap coefficients, a vector of values between 0 and 1, of the same length as the input list.

The cumulative overlap coefficient is calculated by calculating the overlap coefficient of element `i` and the union of elements between 1 and `i-1`. The cumulative overlap coefficient of the first element is set as 0.0.

The cumulative overlap distance is defined in almost the same way, with the only difference the distance is returned. The value of the first element is 1.0. Pratically it is calculated by `1-cumOverlapCoefficient`.

Since the denominator of the overlap coefficient is the size of the smaller set of the two, which is bound to be the size of element `i`, the cumulative overlap distance can be interpreted as the proportion of new items in each new element that are unseen in previous elements. Similarly, the cumulative overlap coefficient can be interpreted as the proportion of items in each new element that have been seen in previous elements. See examples below.

**Note**

An advantage of using cumulative overlap coefficient over cumulative Jaccard Index is that it is monotonic: the value is guaranteed to decrease from 1 to 0, whereas the cumulative Jaccard Index may not be monotonic.
cumsumprop

Examples

```r
cumsumprop(x)
```

```r
myList <- list(first=LETTERS[1:5], second=LETTERS[6:10], third=LETTERS[8:12], fourth=LETTERS[1:12])
cumOverlapCoefficient(myList)
cumOverlapDistance(myList)
```

---

cumsumprop  Proportion of cumulative sum over sum

Description

Proportion of cumulative sum over sum

Usage

```r
cumsumprop(x)
```

Arguments

`x`  Numeric vector

Value

the proportion cumulative sum over sum

Examples

```r
x <- 1:4
cumsumprop(x) # 0.1, 0.3, 0.6, 1
```

---

cutInterval  Cut a vector of numbers into interval factors.

Description

Three types of labels (levels) are supported: “cut.default” (Interval labels returned by cut as default), “left” (Left boundary of intervals), and “right” (Right boundary of intervals).
Usage

cutInterval(
  x,
  step = 1,
  labelOption = c("cut.default", "left", "right"),
  include.lowest = FALSE,
  right = TRUE,
  dig.lab = 3,
  ordered_result = FALSE,
  ...
)

Arguments

  x          A vector of numbers
  step       Step size.
  labelOption How is the label displayed. See details section.
  include.lowest Logical, passed to cut
  right      Logical, passed to cut
  dig.lab    See cut
  ordered_result See cut
  ... Other parameters that are passed to cut

Value

A vector of factors

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

cut

Examples

testNum <- rnorm(100)
(testFac <- cutInterval(testNum, step=1, labelOption="cut.default"))

# compare the result to
(testFacCut <- cut(testNum, 10))
dfFactor

Get a factor vector for a data.frame

Description

The function try to assign a factor vector for a data.frame object. See details below.

Usage

dfFactor(df, sample.group)

Arguments

df A data.frame
sample.group A character, number or a vector of factors, from which the factor vector should be deciphered. See details below.

Details

The function tries to get a factor vector of the same length as the number of rows in the data.frame. The determination is done in the following order: Step 1: It tries to find a column in the data.frame with the name as given by sample.group. If found, this column is transformed into a factor if not and returned. Step 2: It tries to interpret the sample.group as an integer, as the index of the column in the data.frame giving the factor. Step 3: When sample.group itself is a vector of the same length as the data.frame, it is cast to factor when it is still not and returned. Otherwise the program stops with error.

Value

A factor vector with the same length as the data.frame

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

df <- data.frame(gender=c("M", "M", "F", "F", "M"),
age=c(12,12,14,12,14), score=c("A", "B-", "C", "B-", "A"))
dfFactor(df, "gender")
dfFactor(df, "score")
dfFactor(df, 1L)
dfFactor(df, 2L)
dfFactor(df, df$score)
dfFactor2Str  

Convert factor columns in a data.frame into character strings

Description

Convert factor columns in a data.frame into character strings

Usage

dfFactor2Str(df)

Arguments

df  
A data.frame

Value

A data.frame with factor columns coerced into character strings

Examples

exampleDf <- data.frame(Teams=c("HSV", "FCB", "FCB", "HSV"),
        Player=c("Mueller", "Mueller", "Robben", "Holtby"),
        scores=c(3.5, 1.5, 1.5, 1.0), stringsAsFactors=TRUE)
strDf <- dfFactor2Str(exampleDf)
stopifnot(identical(strDf[,1], c("HSV", "FCB", "FCB", "HSV")))
stopifnot(identical(exampleDf[,1], factor(c("HSV", "FCB", "FCB", "HSV"))))

extname

Get the base and extension(s) of file name(s)

Description

Many files have base and extensions in their names, for instance for the file mybook.pdf, the base is mybook and the extension is pdf. basefilename extname functions extract these information from one or more file names.

Usage

extname(x, ifnotfound = NA, lower.case = FALSE)

Arguments

x  
Character vector of file names; other classes will be coerced to characters

ifnotfound  
If no extension name was found, the value to be returned. Default is NA

lower.case  
Logical, should the names returned in lower case?
Value

The base file name or the extension as characters, of the same length as the input file name character. In case that a file name does not contain a extension, NA will be returned.

Note

In case there are multiple dots in the input file name, the last field will be taken as the extension, and the rest as the base name. For instance for file test.out.txt, returned base name is test.out and extension is txt.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

```r
extname("mybook.pdf")
extname("sequence.in.fasta")
extname(c("/path/mybook.pdf", "test.doc"))
extname("README")
extname("README", ifnotfound="")
extname("/path/my\ home/Holiday Plan.txt")

basefilename("mybook.pdf")
basefilename("sequence.in.fasta")
basefilename(c("/path/mybook.pdf", "test.doc"))
basefilename("README")
basefilename("/path/my\ home/Holiday Plan.txt")

basefilename("myBook.pdf", lower.case=TRUE)
extname("myBook.PDF", lower.case=TRUE)
```

Description

Shorten strings to strings with a fix width of characters

Usage

```r
fixWidthStr(str, nchar = 8, align = c("left", "right"))
```
getDefaultFontFamily

Arguments

str       A vector of strings
nchar     The fixed with
align     Character, how to align Strings with more or fewer characters than nchar are either shortened or filled (with spaces)

Value

A vector of strings with fixed widths

Note

NA will be converted to characters and the same fixed width will be applied. The behavior is different from shortenStr, where NA is kept as it is.

See Also

shortenStr

Examples

inputStrs <- c("abc", "abcd", "abcde", "abcdefg", "NA", NA)
outputStrs <- fixWidthStr(inputStrs, nchar=4)
stopifnot(all(nchar(outputStrs)==4))

getDefaultValue

Get default font family

Usage

getDefaultFontFamily()

Value

Character string, the default font family
haltifnot

Ensure the Truth of R Expressions and Print Defined Error Message if NOT

Description
If any of the expressions in `...` are not all TRUE, `stop` is called, producing an error message defined by the `msg` parameter.

Usage
`haltifnot(..., msg = "Error undefined. Please contact the developer")`

Arguments
- `...` any number of ‘logical’ R expressions, which should evaluate to TRUE
- `msg` Error message.

Details
The function is adapted from the `stopifnot` function, with the difference that the error message can be defined by the programmer. With `haltifnot` error message can be more informative, which is desired for diagnostic and user-interation purposes.

Value
NULL if all statements in `...` are TRUE

Author(s)
Jitao David Zhang <jitao_david.zhang@roche.com>

See Also
`stop`, `warning` and `stopifnot`

Examples
```r
haltifnot(!=1, all.equal(pi, 3.14159265), 1<2) ## all TRUE
m <- matrix(c(1,3,3,1), 2,2)
haltifnot(m == t(m), diag(m) == rep(1,2)) ## all TRUE

op <- options(error = expression(NULL))
# "disable stop(.)"  << Use with CARE! >>

haltifnot(all.equal(pi, 3.141593), 2 < 2, all(1:10 < 12), "a" < "b",
        msg="not all conditions are TRUE. Please contact the devleoper")
oneptions(op)# revert to previous error handler
```
Description

These two functions reassembles head and tail, showing the first rows and columns of 2D data structures, e.g. matrix or data.frame.

Usage

headhead(x, m = 6L, n = 6L)

Arguments

x A data.frame or matrix
m Integer, number of rows to show
n Integer, number of columns to show

Details

While head and tail can be applied to data.frame or matrix as well, they show all columns of the first (last) rows even if the matrix has a large number of columns. These two function, headhead and tailtail, circumvent this problem by showing only the first rows AND the first columns.

Value

The first rows/columns of the input object

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

head, tail

Examples

myMat <- matrix(rnorm(10000), nrow=10L)
head(myMat)
headhead(myMat)
tailtail(myMat)
headtail

Print head and tail elements of a vector

Description

This function prints head and tail elements of a vector for visualization purposes. See examples for its usage.

Usage

headtail(vec, head = 2, tail = 1, collapse = ",\", )

Arguments

vec A vector of native types (e.g. character strings)
head Integer, number of head elements to be printed
tail Integer, number of tail elements to be printed
collapse Character string, used to collapse elements

Details

Head and tail elements are concatenated with ellipsis, if there are any elements that are not shown in the vector.

Value

A character string representing the vector

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

head, tail

Examples

testVec1 <- LETTERS[1:10]
headtail(testVec1)
headtail(testVec1, head=3, tail=3)
headtail(testVec1, head=3, tail=3, collapse="|")

testVec2 <- letters[1:3]
headtail(testVec2, head=1, tail=1)
headtail(testVec2, head=2, tail=1)
identicalMatrix

Test whether two matrices are identical by values and by dim names

Description

Test whether two matrices are identical by values and by dim names

Usage

identicalMatrix(x, y, epsilon = 1e-12)

Arguments

x a matrix
y another matrix
epsilon accuracy threshold: absolute differences below this threshold is ignored

Value

Logical

Examples

set.seed(1887); x <- matrix(rnorm(1000), nrow=10, dimnames=list(LETTERS[1:10],NULL))
set.seed(1887); y <- matrix(rnorm(1000), nrow=10, dimnames=list(LETTERS[1:10],NULL))
set.seed(1887); z <- matrix(rnorm(1000), nrow=10, dimnames=list(letters[1:10],NULL))
stopifnot(identicalMatrix(x,y))
stopifnot(!identicalMatrix(x,z))

identicalMatrixValue

Test whether two matrices have the same numeric values given certain accuracy

Description

Test whether two matrices have the same numeric values given certain accuracy

Usage

identicalMatrixValue(x, y, epsilon = 1e-12)
**imatch**

Arguments

- `x` a matrix
- `y` another matrix
- `epsilon` accuracy threshold: absolute differences below this threshold is ignored

Value

Logical

Examples

```r
set.seed(1887); x <- matrix(rnorm(1000), nrow=10)
set.seed(1887); y <- matrix(rnorm(1000), nrow=10)
set.seed(1882); z <- matrix(rnorm(1000), nrow=10)
stopifnot(identicalMatrixValue(x,y))
stopifnot(!identicalMatrixValue(x,y+1E-5))
stopifnot(!identicalMatrixValue(x,y-1E-5))
stopifnot(!identicalMatrixValue(x,z))
```

---

**imatch**

*Case-insensitive match and pmatch*

Description

Case-insensitive match and pmatch functions, especially useful in parsing user inputs, e.g. from command line.

Usage

```r
imatch(x, table, ...)
```

Arguments

- `x` String vector
- `table` A vector to be matched
- `...` Other parameters passed to match or pmatch

Details

`imatch` and `ipmatch` work similar as `match` and `pmatch`, except that they are case-insensitive. `matchv`, `imatchv` and `ipmatchv` are shots-cuts to get the matched value (therefore the ‘v’) if the match succeeded, or NA if not. `match(x,table)` is equivalent to `table[[match(x,table)]]`. See examples.
Value

imatch and ipmatch returns matching indices, or NA (by default) if the match failed.
matchv, imatchv and ipmatchv returns the matching element in table, or NA if the match failed.
Note that when cases are different in x and table, the one in table will be returned. This is especially useful for cases where user’s input has different cases as the internal options.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

match and pmatch

Examples

user.input <- c("hsv", "BvB")
user.input2 <- c("HS", "BV")
internal.options <- c("HSV", "FCB", "BVB", "FCN")

match(user.input, internal.options)
imatch(user.input, internal.options)
ipmap(user.input, internal.options)
ipmap(user.input2, internal.options)

matchv(user.input, internal.options)
matchv(tolower(user.input), tolower(internal.options))
imatchv(user.input, internal.options)
ipmap(user.input, internal.options)
ipmap(user.input2, internal.options)

isDir

Checks existing directory

Description

Checks whether given character strings point to valid directories

Usage

isDir(...)

checkDir(...)

assertDir(...)
isError

Arguments
... One or more character strings giving directory names to be tested

Details
isDir tests whether the given string represent a valid, existing directory. assertDir performs a logical test, and stops the program if the given string does not point to a given directory.
checkDir is synonymous to isDir

Value
isDir returns logical vector.
assertDir returns an invisible TRUE if directories exist, otherwise halts and prints error messages.

Author(s)
Jitao David Zhang <jitao_david.zhang@roche.com>

See Also
file.info, checkFile and assertFile

Examples

dir1 <- tempdir()
dir2 <- tempdir()

isDir(dir1, dir2)
assertDir(dir1, dir2)

isError  Tell whether an object is an error

Description
Determines whether an object is of class try-error

Usage
isError(x)

Arguments
x Any object, potentially produced within a try-error structure.
isRocheCompoundID

Value

 Logical value, TRUE if x inherits the try-error class.

Author(s)

 Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

```r
if(exists("nonExistObj")) rm(nonExistsObj)
myObj <- try(nonExistObj/5, silent=TRUE)
isError(myObj)
```

isRocheCompoundID

Tell whether a character string is a Roche compound ID

Description

Tell whether a character string is a Roche compound ID

Usage

```r
isRocheCompoundID(str)
```

Arguments

```r
str Character string(s)
```

Value

 A logical vector of the same length as str, indicating whether each element is a Roche compound ID or not.

 Short versions (RO[1-9]2,7) are supported.

Examples

```r
isRocheCompoundID(c("RO1234567", "RO-1234567",
"RO1234567-000", "RO1234567-000-000",
"ROnoise-000-000"))
```
### jaccardIndex

*Calculate the Jaccard Index between two vectors*

**Description**

Calculate the Jaccard Index between two vectors

**Usage**

```r
jaccardIndex(x, y)
```

```r
jaccardDistance(x, y)
```

**Arguments**

- `x`: A vector
- `y`: A vector

**Value**

The Jaccard Index, a number between 0 and 1

JaccardDistance is defined as $1 - \text{JaccardIndex}$.

**Examples**

```r
myX <- 1:6
myY <- 4:9
jaccardIndex(myX, myY)
jaccardDistance(myX, myY)

myX <- LETTERS[1:5]
myY <- LETTERS[6:10]
jaccardIndex(myX, myY)
jaccardDistance(myX, myY)
```

---

### keepMaxStatRow

*KEEP ROWS WITH THE MAXIMUM STATISTIC*

**Description**

A common task in expression analysis is to collapse multiple features that are mapped to the same gene by some statistic. This function does this job by keeping the matrix row (normally features) with the highest statistic specified by the user.
Usage

keepMaxStatRow(
  matrix,
  keys,
  keepNArows = TRUE,
  stat = function(x) mean(x, na.rm = TRUE),
  levels = c("rownames", "attribute", "discard"),
  ...
)

Arguments

matrix
  A numeric matrix

keys
  A vector of character giving the keys the rows are mapped to. A common sce-
  nario is that each row represents one probeset, while the vector keys give the
  genes that the probesets are mapped to. Thus keys can be redundant, namely
  multiple probesets can map to the same gene.

keepNArows
  Logical, whether rows with NA as their keys should be kept (TRUE) or should be
  discarded (FALSE)

stat
  The function to calculate the univariate statistic. By default the NA-robust mean
  is used.

levels
  How should the information of the levels of keys, e.g. unique keys, be kept. dicard
  will discard this information, rownames will make the unique keys (poten-
  tially with NAs) as row names of the output matrix, and attribute will ap-
  pend an attribute named levels to the output matrix.

...
  Other parameters passed to the stat function

Details

isMaxStatRow returns a logical vector, with rows with maximal statistics each key as TRUE and oth-
  erwise as FALSE. keepMaxStatRowInd returns the integer indices of such rows. Finally keepMaxStatRow
  returns the resulting matrices.

For use see examples

Value

A numeric matrix with rows mapped to unique keys, selected by the maximum statistics. See
  examples below

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>
libordie

Load a library mutedly and quit (die) in case of failing

Description

The specified library is loaded mutedly by suppressing all messages. If the library is not found, or its version under the specification of minVer, the R session dies with a message.
Usage

libordie(package, minVer, missing.quit.status = 1, ver.quit.status = 1)

Arguments

package One package name (can be character or non-quoted symbol (see examples)
minVer Optional, character string, the minimum working version
missing.quit.status Integer, the status of quitting when the package was not found
ver.quit.status Integer, the status of quitting when the package was found, but older than the minimum working version

Details

Only one package should be tested once.

Value

NULL if success, otherwise the session will be killed.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

The function calls qqmsg internally to kill the session

Examples

if(interactive()) {
  libordie(stats)
  libordie("methods")
  libordie(base, minVer="2.15-1")
}
list2df

Transform a list of character strings into a data.frame

Description

Transform a list of character strings into a data.frame

Usage

list2df(list, names = NULL, col.names = c("Name", "Item"))

Arguments

- list: A list of character strings
- names: Values in the 'Name' column of the result, used if the input list has no names
- col.names: Column names of the data.frame

Value

A data.frame

Examples

myList <- list(HSV=c("Mueller", "Papadopoulos", "Wood"), FCB=c("Lewandowski", "Robben", "Hummels"), BVB=c("Reus", "Goetze", "Kagawa"))
list2df(myList, col.names=c("Club", "Player"))

listOverlapCoefficient

Pairwise overlap coefficient of lists

Description

Pairwise overlap coefficient of lists

Usage

listOverlapCoefficient(x, y = NULL, checkUniqueNonNA = TRUE)
Arguments

- **x**: A list of vectors that are interpreted as sets of elements.
- **y**: A list of vectors that are interpreted as sets of elements. In case of NULL, pairwise overlap coefficient of lists in x is returned.
- **checkUniqueNonNA**: Logical, should vectors in the list be first cleaned up so that NA values are removed and the elements are made unique? Default is set as TRUE; if the user is confident that the vectors are indeed valid sets, this option can be set as FALSE to speed up the code.

Value

A matrix of column-wise pairwise overlap coefficients.

Examples

```r
set.seed(1887)
testSets1 <- sapply(rbinom(10, size=26, prob=0.3),
  function(x) sample(LETTERS, x, replace=FALSE))
names(testSets1) <- sprintf("List%d", seq(along=testSets1))
testSets1Poe <- listOverlapCoefficient(testSets1)
testSets1PoeNoCheck <- listOverlapCoefficient(testSets1, checkUniqueNonNA=FALSE)
stopifnot(identical(testSets1Poe, testSets1PoeNoCheck))
testSets2 <- sapply(rbinom(15, size=26, prob=0.3),
  function(x) sample(LETTERS, x, replace=FALSE))
names(testSets2) <- sprintf("AnotherList%d", seq(along=testSets2))
testSets12Poe <- listOverlapCoefficient(testSets1, testSets2)
```

---

### longdf2matrix

Convert a long-format data frame into matrix

**Description**

Input data.frame must contain at least three columns: one contains row names (specified by `row.col`), one contains column names (`column.col`), and one contains values in matrix cells (`value.col`). The output is a 2D matrix.

**Usage**

```r
longdf2matrix(
  df, 
  row.col = 1L, 
  column.col = 2L, 
  value.col = 3L, 
  missingValue = NULL
)
```
**matchColumn**

**Arguments**

- `df` (Long-format data frame)
- `row.col` (Character or integer, which column of the input data.frame contains row names?)
- `column.col` (Character or integer, which column contains column names?)
- `value.col` (Character or integer, which column contains matrix values?)
- `missingValue` (Values assigned in case of missing data)

**Value**

A 2D matrix equivalent to the long-format data frame

**Author(s)**

Jitao David Zhang <jitao_david.zhang@roche.com>

**See Also**

`matrix2longdf`

**Examples**

```r
test.df <- data.frame(H=c("HSV", "BVB", "HSV", "BVB"),
  A=c("FCB", "S04", "S04", "FCB"),
  score=c(3, 1, 1, 0))
longdf2matrix(test.df, row.col=1L, column.col=2L, value.col=3L)
data(Indometh)
longdf2matrix(Indometh, row.col="time", column.col="Subject", value.col="conc")
longdf2matrix(Indometh, row.col="Subject", column.col="time", value.col="conc")
```

**Description**

Given a vector known as master vector, a data.frame and one column of the data.frame, the function `matchColumnIndex` matches the values in the column to the master vector, and returns the indices of each value in the column with respect to the vector. The function `matchColumn` returns whole or subset of the data.frame, with the matching column in the exact order of the vector.

**Usage**

```r
matchColumn(vector, data.frame, column, multi = FALSE)
```
**Arguments**

- **vector**
  - A vector, probably of character strings.

- **data.frame**
  - A data.frame object

- **column**
  - The column name (character) or index (integer between 1 and the column number), indicating the column to be matched. Exceptionally 0 is as well accepted, which will match the row names of the data.frame to the given vector.

- **multi**
  - Logical, deciding what to do if a value in the vector is matched to several values in the data.frame column. If set to TRUE, all rows containing the matched value in the specified column are returned; otherwise, when the value is set to FALSE, one arbitrary row is returned. See details and examples below.

**Details**

See more details below.

The function is used to address the following question: how can one order a data.frame by values of one of its columns, the order for which is given in a vector (known as “master vector”). `matchColumnIndex` and `matchColumn` provide thoroughly-tested implementation to address this question.

For one-to-one cases, where both the column and the vector have no duplicates and can be matched one-to-one, the question is straightforward to solve with the `match` function in R. In one-to-many or many-to-many matching cases, the parameter `multi` determines whether multiple rows matching the same value should be shown. If `multi=FALSE`, then the sorted data.frame that are returned has exactly the same row number as the input vector; otherwise, the returned data.frame has more rows. See the examples below.

In either case, in the returned data.frame object by `matchColumn`, values in the column used for matching are overwritten by the master vector. If `multi=TRUE`, the order of values in the column is also obeying the order of the master vector, with exceptions of repeating values caused by multiple matching.

The `column` parameter can be either character string or non-negative integers. In the exceptional case, where `column=0L` (“L” indicates integer), the row names of the data.frame is used for matching instead of any of the columns.

Both functions are NA-friendly, since NAs in neither vector nor column should break the code.

**Value**

For `matchColumnIndex`, if `multi` is set to FALSE, an integer vector of the same length as the master vector, indicating the order of the data.frame rows by which the column can be re-organized into the master vector. When `multi=TRUE`, the returning object is a list of the same length as the master vector, each item containing the index (indices) of data.frame rows which match to the master vector.

For `matchColumn`, a data.frame is always returned. In case `multi=FALSE`, the returning data frame has the same number of rows as the length of the input master vector, and the column which was specified to match contains the master vector in its order. If `multi=TRUE`, returned data frame can contain equal or more numbers of rows than the master vector, and multiple-matched items are repeated.
matchColumnName

Author(s)
Jitao David Zhang <jitao_david.zhang@roche.com>

See Also
See match for basic matching operations.

Examples

def <- data.frame(Team=c("HSV", "BVB", "HSC", "FCB", "HSV"),
Pkt=c(25,23,12,18,21),
row.names=c("C", "B", "A", "F", "E"))
teams <- c("HSV", "BVB", "BRE", NA)

matchColumnIndex(teams, df, 1L, multi=FALSE)
matchColumnIndex(teams, df, 1L, multi=TRUE)
matchColumnIndex(teams, df, "Team", multi=FALSE)
matchColumnIndex(teams, df, "Team", multi=TRUE)
matchColumnIndex(teams, df, 0, multi=FALSE)
matchColumnIndex(ind, df, 0, multi=FALSE)
matchColumnIndex(ind, df, 0, multi=TRUE)

matchColumn(teams, df, 1L, multi=FALSE)
matchColumn(teams, df, 1L, multi=TRUE)
matchColumn(teams, df, "Team", multi=FALSE)
matchColumn(teams, df, "Team", multi=TRUE)
matchColumn(ind, df, 0, multi=FALSE)
matchColumn(ind, df, 0, multi=TRUE)

matchColumnName

Match a given vector to column names of a data.frame or matrix

Description
Match a given vector to column names of a data.frame or matrix

Usage
matchColumnName(data.frame.cols, reqCols, ignore.case = FALSE)

Arguments
data.frame.cols
column names of a data.frame. One can also provide a data.frame, which may however cause worse performance since the data.frame is copied
reqCols
required columns
ignore.case
logical, whether the case is considered
matrix2longdf

Transform a matrix into a long-format data.frame

Description

The function converts a matrix into a long-format, three-column data.frame, containing row, column names, and value. Such ‘long’ data.frames can be useful in data visualization and modelling.

Usage

matrix2longdf(
  mat,
  row.names,
  col.names,
  longdf.colnames = c("row", "column", "value")
)

Arguments

mat
A matrix

row.names
Character, row names to appear in the data.frame. If missing, the rownames of the matrix will be used. If set to NULL, or if the matrix rownames are NULL, a integer index vector starting from 1 will be used.

col.names
Character, column names to appear in the data.frame. The rule of handling missing or NULL parameters is the same as row.names described above.

longdf.colnames
Character, column names of the output long data frame

Details

The function converts a matrix into a three-column, ‘long’ format data.frame containing row names, column names, and values of the matrix.
mergeInfreqLevelsByCumsumprop

**Value**

A data.frame object with three columns: row, column and value. If the input matrix is of dimension MxN, the returning data.frame is of the dimension MNx3.

**Note**

The length of row.names and col.names should be as the same as the matrix dimension. Otherwise the function raises warnings.

**Author(s)**

Jitao David Zhang <jitao_david.zhang@roche.com>

**Examples**

```r
test.mat <- matrix(1:12, ncol=4, nrow=3, dimnames=list(LETTERS[1:3], LETTERS[1:4]))
print(test.mat)
print(matrix2longdf(test.mat))
print(matrix2longdf(test.mat, longdf.colnames=c("From", "To", "Time")))
```

---

**Description**

Merge infrequent levels by setting the threshold of the proportion of cumulative sum over sum a.k.a. cumsumprop

**Usage**

```r
mergeInfreqLevelsByCumsumprop(
  classes,
  thr = 0.9,
  mergedLevel = "others",
  returnFactor = TRUE
)
```

**Arguments**

- **classes**: Character strings or factor.
- **thr**: Numeric, between 0 and 1, how to define frequent levels. Default: 0.9, namely levels which make up over 90% of all instances.
- **mergedLevel**: Character, how the merged level should be named.
- **returnFactor**: Logical, whether the value returned should be coerced into a factor.
Value

A character string vector or a factor, of the same length as the input classes, but with potentially fewer levels.

Note

In case only one class is deemed as infrequent, its label is unchanged.

Examples

```r
set.seed(1887)
myVals <- sample(c(rep("A", 4), rep("B", 3), rep("C", 2), "D"))
## in the example below, since A, B, C make up of 90% of the total, 
## D is infrequent. Since it is alone, it is not merged
mergeInfreqLevelsByCumsumprop(myVals, 0.9)
mergeInfreqLevelsByCumsumprop(myVals, 0.9, returnFactor=FALSE) ## return characters 
## in the example below, since A and B make up 70% of the total, 
## and A, B, C 90%, they are all frequent and D is infrequent. 
## Following the logic above, no merging happens
mergeInfreqLevelsByCumsumprop(myVals, 0.8)
mergeInfreqLevelsByCumsumprop(myVals, 0.7) ## A and B are left, C and D are merged
mergeInfreqLevelsByCumsumprop(myVals, 0.5) ## A and B are left, C and D are merged
mergeInfreqLevelsByCumsumprop(myVals, 0.4) ## A is left
mergeInfreqLevelsByCumsumprop(myVals, 0.3) ## A is left
```

midentical  Multiple identical

Description

Testing whether multiple objects are identical

Usage

```r
midentical(
  ..., 
  num.eq = TRUE, 
  single.NA = TRUE, 
  attrib.as.set = TRUE, 
  ignore.bytecode = TRUE, 
  ignore.environment = FALSE, 
  ignore.srcref = TRUE
)
```
**mmatch**

| 43 |

**Arguments**

... Objects to be tested, or a list of them
num.eq, single.NA, attrib.as.set, ignore.bytecode,
See **identical**
ignore.environment, ignore.srcref
See **identical**

**Details**

`midentical` extends `identical` to test multiple objects instead of only two.

**Value**

A logical value, `TRUE` if all objects are identical

**Author(s)**

Jitao David Zhang <jitao_david.zhang@roche.com>

**See Also**

`identical`

**Examples**

```r
set1 <- "HSV"
set2 <- set3 <- set4 <- c("HSV", "FCB")

midentical(set1, set2)
midentical(list(set1, set2))

midentical(set2, set3, set4)
midentical(list(set2, set3, set4))

## other options passed to identical
midentical(0, -0, +0, num.eq=FALSE)
midentical(0, -0, +0, num.eq=TRUE)
```

**Description**

Multiple matching between two vectors. Different from R-native `match` function, where only one match is returned even if there are multiple matches, `mmatch` returns all of them.
Usage

\texttt{mmatch(x, table, nomatch = NA\_integer\_)}

Arguments

- \texttt{x}: vector or \texttt{NULL}: the values to be matched.
- \texttt{table}: vector or \texttt{NULL}: the values to be matched against.
- \texttt{nomatch}: the value to be returned in case when no match is found.

Details

Multiple matches can be useful in many cases, and there is no native R function for this purpose. User can write their own functions combining \texttt{lapply}ing with \texttt{match} or \texttt{%in%}, our experience however shows that such non-vectorized function can be extremely slow, especially when the \texttt{x} or \texttt{table} vector gets longer.

\texttt{mmatch} delegates the multiple-matching task to a C-level function, which is optimized for speed. Internal benchmarking shows improvement of hundred fold, namely using \texttt{mmatching} costs about 1/100 of the time used by R-implementation.

Value

A list of the same length as the input \texttt{x} vector. Each list item contains the matching indices (similar to \texttt{match}).

Author(s)

Jitao David Zhang <jitao\_david.zhang@roche.com>, C-code was adapted from the program written by Roland Schmucki.

See Also

\texttt{match}

Examples

\begin{verbatim}
vec1 <- c("HSV", "BVB", "FCB", "HSV", "BRE", "HSV", NA, "BVB")
vec2 <- c("FCB", "FCN", "FCB", "HSV", "BVB", "HSV", "FCK", NA, "BRE", "BRE")

mmatch(vec1, vec2)

## compare to match
match(vec1, vec2)
\end{verbatim}
**munion**  

*Operations for multiple sets*

**Description**

Set operation functions in the base package, `union`, `intersect` and `setdiff`, can only be applied to binary manipulations involving two sets. Following functions, `munion`, `mintersect` and `msetdiff`, extend their basic versions to deal with multiple sets.

**Usage**

```r
munion(...)
```

**Arguments**

...  

Vectors of items, or a list of them. See examples below.

**Details**

These functions apply set manipulations (union, intersect, or difference) in a sequential manner: the first two sets are considered first, then the third, the fourth and so on, till all sets have been visited.

**Value**

A vector of set operation results. Can be an empty vector if no results were returned.

**Author(s)**

Jitao David Zhang <jitao_david.zhang@roche.com>

**See Also**

`union`, `intersect` and `setdiff`.

**Examples**

```r
set1 <- c("HSV", "FCB", "BVB", "FCN", "HAN")
set2 <- c("HSV", "FCB", "BVB", "HAN")
set3 <- c("HSV", "BVB", "FSV")

munion(set1, set2, set3)
mintersect(set1, set2, set3)
msetdiff(set1, set2, set3)

## sets can be given in a list as well
munion(list(set1, set2, set3))
mintersect(list(set1, set2, set3))
msetdiff(list(set1, set2, set3))
```
### na.false

**Replace NA with FALSE**

**Description**

Replace NA in a vector with FALSE

**Usage**

```r
na.false(x)
```

**Arguments**

- `x` A logical vector or matrix

**Value**

Logical vector or matrix with NAs replaced by FALSE

**Author(s)**

Jitao David Zhang &lt;jitao_david.zhang@roche.com&gt;

**See Also**

```r
myX <- c("HSV", "FCK", "FCN", NA, "BVB") res <- myX == "HSV" na.false(res)
```

### naivePairwiseDist

**Calculate pairwise distances between each pair of items in a list**

**Description**

Calculate pairwise distances between each pair of items in a list

**Usage**

```r
naivePairwiseDist(list, fun = jaccardIndex)
```

**Arguments**

- `list` A list
- `fun` A function that receives two vectors (such as jaccardIndex) and returns a number (scale)
Value

A symmetric matrix of dimension $m \times m$, where $m$ is the length of the list

This function is inefficient compared with matrix-based methods. It is exported just for education and for verifying results of matrix-based methods.

Examples

```r
myList <- list(first=LETTERS[3:5], second=LETTERS[1:3], third=LETTERS[1:5], fourth=LETTERS[6:10])
naivePairwiseDist(myList, fun=jaccardIndex)
## despite of the name, any function that returns a number can work
naivePairwiseDist(myList, fun=jaccardDistance)
```

---

**ofactor**

*Ordered factor*

Description

Build a factor using the order of input character strings

Usage

```r
ofactor(x, ...)
```

Arguments

- `x` A vector of character strings
- `...` Other parameters passed to `factor`

Value

Factor with levels in the same order of the input strings.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

`factor`
Examples

testStrings <- c("A", "C", "B", "B", "C")
(testFac <- factor(testStrings))
(testOfac <- ofactor(testStrings))

stopifnot(identical(levels(testOfac), c("A", "C", "B")))

desc

openFileDevice

Open a device as a file preparing for plotting in the file

Description

The function openFileDevice opens a device of the type specified by the file extension name. It such prepares the file for visualizing data. User must call dev.off once the writing (plotting) to the device is finished.

Usage

openFileDevice(filename, width = 7, height = 7, dpi = 300L, family)

Arguments

filename Character, file name to be written to. The type of file is determined by the extension. See details below.

width Number, figure width of the file in inch.

height Number, figure height of the file in inch.

dpi Number, resolution as “dots per inch”. For publication 300dpi is usually enough.

family Font family name. Only applicable to PDF files

Details

closeFileDevice quietly closes the current device: it does not print the information of the next device.

The function openFileDevice calls extname to determine the file type to be drawn in. Currently supported types include PDF, tiff (tif), bmp, jpeg (jpeg). When the file type is not recognized, the PDF format is used as a fallback.

As an example, myplot.pdf will triggers openning a PDF device, newplot.png a PNG device, and oldplot.tif a TIFF device, whereas myfile.abc will be openned as a PDF device.

For bitmap files like BMP, JPEG,PNG and TIFF, we use inch as the size unit in order to be compatible with PDF. And the resolution is always set to 300dpi. Furthermore, JPEG quality is set to 90 instead of the default value 75, and TIFF do not use any compression. These settings follow our practices for scientific publication while allowing generic post-precessing of figures.
overlapCoefficient

Value

Both functions are used for its side effect.

Note

After plotting, user should call dev.off to close the device in the file, otherwise the file can probably not be read.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

extname for getting extension name of file. See pdf, png, jpeg, tiff and bmp for file formats.

Examples

if(interactive()) {
  tempfile1 <- paste(tempfile(), ".pdf", sep="")
  openFileDevice(tempfile1)
  plot(rnorm(100), rnorm(100))
  closeFileDevice()

  tempfile2 <- paste(tempfile(), ".png", sep="")
  openFileDevice(tempfile2, width=5, height=5)
  plot(rnorm(100), rnorm(100))
  closeFileDevice()
}

overlapCoefficient  Overlap coefficient, also known as Szymkiewicz-Simpson coefficient

Description

Overlap coefficient, also known as Szymkiewicz-Simpson coefficient

Usage

overlapCoefficient(x, y, checkUniqueNonNA = FALSE)

overlapDistance(x, y, checkUniqueNonNA = FALSE)
Arguments

x  A vector
y  A vector
checkUniqueNonNA
   Logical, if TRUE, x and y are made unique and non-NA

Value

The overlap coefficient

See Also

jaccardIndex

overlapCoefficient calculates the overlap coefficient, and overlapDistance is defined by 1-overlapCoefficient.

Examples

myX <- 1:6
myY <- 4:9
overlapCoefficient(myX, myY)

myY2 <- 4:10
overlapCoefficient(myX, myY2)
## compare the result with Jaccard Index
jaccardIndex(myX, myY2)

## overlapDistance
overlapDistance(myX, myY2)

---

**pAbsLog10Score**  Transform p-values to continuous scores with the absolute-log10 transformation

Description

The function maps p values between 0 and 1 to continuous scores ranging on R by the following equation: \( \text{abs}(\log_{10}(p)) \times \text{sign} \)

Usage

pAbsLog10Score(p, sign = 1)
pairwiseJaccardIndex

Arguments

p                      p-value(s) between (0,1]
sign                  Sign of the score, either positive (in case of positive numbers), negative (in case of negative numbers), or zero. In case a logical vector, TRUE is interpreted as positive and FALSE is interpreted as negative.

See Also

pQnormScore, pScore

Examples

testPvals <- c(0.001, 0.01, 0.05, 0.1, 0.5, 1)
pAbsLog10Score(testPvals)
testPvalSign <- rep(c(-1,1), 3)
pAbsLog10Score(testPvals, sign=testPvalSign)
testLog <- rep(c(TRUE, FALSE),3)
pAbsLog10Score(testPvals, testLog)

pairwiseJaccardIndex  Calculate pairwise Jaccard Indices between each pair of items in a list

Description

Calculate pairwise Jaccard Indices between each pair of items in a list

Usage

pairwiseJaccardIndex(list)

pairwiseJaccardDistance(list)

Arguments

list              A list

Value

A symmetric matrix of dimension m\times m, where \(m\) is the length of the list pairwiseJaccardDistance is defined as 1-pairwiseJaccardIndex.
Examples

```r
myList <- list(first=LETTERS[3:5], second=LETTERS[1:3], third=LETTERS[1:5], fourth=LETTERS[6:10])
pairwiseJaccardIndex(myList)

poormanPJI <- function(list) {
  sapply(list, function(x) sapply(list, function(y) jaccardIndex(x,y)))
}
stopifnot(identical(pairwiseJaccardIndex(myList), poormanPJI(myList))
```

Description

Calculate pairwise overlap coefficients between each pair of items in a list

Usage

```r
pairwiseOverlapDistance(list)
pairwiseOverlapCoefficient(list)
```

Arguments

- `list` A list

Value

A symmetric matrix of dimension $m \times m$, where $m$ is the length of the list. `pairwiseOverlapDistance` is defined the pairwise overlap distance.

Examples

```r
myList <- list(first=LETTERS[3:5], second=LETTERS[1:3], third=LETTERS[1:5], fourth=LETTERS[6:10])
pairwiseOverlapCoefficient(myList)
pairwiseOverlapDistance(myList)

poormanPOC <- function(list) {
  sapply(list, function(x) sapply(list, function(y) overlapCoefficient(x,y)))
}
stopifnot(identical(pairwiseOverlapCoefficient(myList), poormanPOC(myList)))
```
percentage

Print a decimal number in percent format

Description
Print a decimal number in percent format

Usage
percentage(x, fmt = "1.1")

Arguments
x a decimal number, usually between -1 and 1
fmt format string, '1.1' means a digit before and after the decimal point will be printed

Value
Character string

Examples
percentage(c(0,0.1,0.25,1))
percentage(c(0,0.1,0.25,1), fmt="1.4")
percentage(c(0,-0.1,0.25,-1), fmt="+1.1")

pQnormScore

Transform p-values to continuous scores with the quantile function of the normal distribution

Description
Quantile function, also known as the inverse of cumulative distribution function of the normal distribution, is used to map p-values to continuous scores raging on R. The signs of the resulting scores are positive by default and are determined by the parameter sign.

Usage
pQnormScore(p, sign = 1)
Arguments

\( p \) \hspace{1em} \text{\( p \)-value(s) between (0, 1]} \\
\( \text{sign} \) \hspace{1em} \text{Signs of the scores, either positive (in case of positive numbers), negative (in case of negative numbers), or zero. In case of a logical vector, TRUE is interpreted as positive and FALSE is interpreted as negative.}

See Also

\texttt{pAbsLog10Score}, \texttt{pScore}

Examples

```r
testPvals <- c(0.001, 0.01, 0.05, 0.1, 0.5, 1)
pQnormScore(testPvals)

testPvalSign <- rep(c(-1,1), 3)
pQnormScore(testPvals, sign=testPvalSign)

testLog <- rep(c(TRUE, FALSE),3)
pQnormScore(testPvals, testLog)
```

print.BEDAinfo \hspace{1em} \textit{Print BEDAinfo object}

Description

Print BEDAinfo object

Usage

```r
## S3 method for class 'BEDAinfo'
print(x, ...)  
```

Arguments

\( x \) \hspace{1em} \text{A BEDA info object, returned by \texttt{bedaInfo}} \\
\( \ldots \) \hspace{1em} \text{Ignored}

Value

\text{Invisible NULL, only side effect is used}

Examples

```r
if(interactive()) {print(bedaInfo())}
```
**pScore**

Transform *p*-values to continuous scores

**Description**

The function wraps other functions to map *p* values ranging on *(0,1]* to continuous scores ranging on \( \mathbb{R} \) in a number of ways.

**Usage**

```r
pScore(p, sign = 1, method = c("qnorm", "absLog10"))
```

**Arguments**

- **p**: *p*-value between *(0,1]*
- **sign**: Sign of the score, either positive (in case of positive numbers), negative (in case of negative numbers), or zero. In case a logical vector, TRUE is interpreted as positive and FALSE is interpreted as negative.
- **method**: Currently available methods include qnorm and absLog10.

**See Also**

- `pAbsLog10Score`
- `pQnormScore`

**Examples**

```r
testPvals <- c(0.001, 0.01, 0.05, 0.1, 0.5, 1)
pScore(testPvals, method="absLog10")
pScore(testPvals, method="qnorm")
testPvalSign <- rep(c(-1,1), 3)
pScore(testPvals, sign=testPvalSign, method="absLog10")
pScore(testPvals, sign=testPvalSign, method="qnorm")
testLog <- rep(c(TRUE, FALSE),3)
pScore(testPvals, testLog, method="absLog10")
pScore(testPvals, testLog, method="qnorm")

testPvals <- 10^seq(-5, 0, 0.05)
plot(pScore(testPvals, method="qnorm"),
     pScore(testPvals, method="absLog10"),
     xlab="pQnormScore", ylab="pAbsLog10Score"); abline(0,1, col="red", lty=2)
```
**putColsFirst**  
*Rearrange columns to put some columns to far left*

**Description**  
This function is helpful to export tables where certain columns are desired to be placed to the most left of the data.frame

**Usage**  
```r
putColsFirst(data.frame, columns)
```

**Arguments**  
- `data.frame`: Data.frame
- `columns`: Character vector, names of columns which are to be put to the left

**Value**  
data.frame with re-arranged columns

**Author(s)**  
Jitao David Zhang <jitao_david.zhang@roche.com>

**Examples**

```r
clubs <- data.frame(Points=c(21,23,28,24), Name=c("BVB", "FCB", "HSV", "FCK"), games=c(12,11,11,12))
putColsFirst(clubs, c("Name"))
putColsFirst(clubs, c("Name", "games"))
```

---

**pwdecode**  
*Decode password with function implemented with pwencode*

**Description**  
Decode password encrypted with pwencode.

**Usage**  
```r
pwdecode(password)
```
Arguments

password Character string to be decoded. If starting with a empty character, the string is sent for decoding; otherwise, it is deemed as clear text password and returned.

Details

See `pwdecode` function documentation in BIOS for implementation details.

Note that since R does not support strings embedding null values (\000), the password to be decoded has to be given with two slashes, e.g. ‘\001\000\129\235’.

Value

Decoded character string, or empty string if decoding fails

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>. The C library code was written by Detlef Wolf.

Examples

```r
mycode <- " \001\000\141\314\033\033\033\033\033\142\303\056\166\311\037\042"
pwdecode(mycode)
```

---

**pwencode**

Encode a password

Description

Encode a password

Usage

```r
pwencode(label = "VAR", key)
```

Arguments

- **label** label used to encode the password
- **key** password key

Value

Character string, encoded password
qqmsg

Quiteley Quit with Messages

Description

Quiteley quit R with messages in non-interactive sessions

Usage

qqmsg(..., status = 0, save = FALSE, runLast = TRUE)

Arguments

... Messages to be passed to message
status Quit stats
save Logical, should current working environment be saved?
runLast Logical, should .Last() be executed?

Details

The function prints messages in any case, and quits R if the current session is non-interactive, e.g. in the command-line running Rscript mode

Value

Invisible NULL, only side effect is used.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

quit

Examples

## the example should not run because it will lead the R session to quit
## Not run:
qqmsg()
qqmsg("die", status=0)
qqmsg("Avada kedavra", status=-1)
qqmsg("Crucio!", "\n", "Avada kedavra", status=-100)

## End(Not run)
qsystem

Quietly runs a system command

Description

Quietly runs a system command: the output is internalized and returned as an invisible variable, and the standard error output is ignored.

Usage

qsystem(command)

Arguments

command A system command

Details

The function runs the system command in a quiet mode. The function can be useful in CGI scripts, for instance

Value

(Invisibly) the internalized output of the command

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

dateIntern <- system("date")

refactorNum

Sort numeric factor levels by values

Description

Factor variables with numbers as levels are alphabetically ordered by default, which requires rearrangements for various purposes, e.g. modelling or visualizations. This function re-orders levels of numeric factor variables numerically.

Usage

refactorNum(x, decreasing = FALSE)
registerLog

Arguments

x A factor variable with numeric values as levels
decreasing Logical, should the levels sorted descendingly?

Value

A factor variable, with sorted numeric values as levels

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

(nums <- factor(c("2","4","24","1","2","125","1","2","125")))
(nums.new <- refactorNum(nums))

registerLog

The functions registerLog and doLog provide a simple mechanism to handle loggings (printing text messages to files or other types of connections) in R.

Description

Users can register arbitrary numbers of loggers with registerLog, and the functions take care of low-level details such as openning and closing the connections.

Usage

registerLog(..., append = FALSE)

Arguments

... Arbitrary numbers of file names (character strings) or connection objects (see example).
append Logical, log will be appended to the existing file but not overwriting. Only valid for files but not for connections such as standard output.
Details

Input parameters can be either character strings or connections (such as the objects returned by `stdout()` or `pipe()`).

If a character string is registered as a logger, it is assumed as a file name (user must make sure that it is writable/appendable). In case the file exists, new logging messages will be appended; otherwise if the file does not exist, it will be created and the logging messages will be written to the file.

A special case is the parameter value "-": it will be interpreted as standard output.

If a connection is registered as a logger, it must be writable in order to write the logging messages.

Each parameter will be converted to a connection object, which will be closed (when applicable) automatically before R quits.

If the parameter is missing (or set to NA or NULL), no logging will take place.

Value

No value returned: its side effect is used.

Note

Currently, the loggers are stored in a variable in the namespace of ribiosUtils named RIBIOS_LOGGERS. This is only for internal use of the package and may change any time, therefore users are not advised to manipulate this variable directly.

To clear the registered loggers, use `clearLog`. To flush the registered loggers, use `flushLog`. Usually it is not necessary to use `flushLog` in R scripts, since by program exit the active R session will automatically flush and close the connections (in addition, frequent flushing may decrease the program’s efficiency). However, if used in interactive sessions, sometimes `flushLog` is needed to force R write all log files to all connections that are registered.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

doLog writes messages iteratively to each connection registered by `registerLog`.

Examples

```r
logfile1 <- tempfile()
logfile2 <- tempfile()
logcon3 <- stdout()
if(.Platform$OS.type == "unix") {
  registerLog("/dev/null")
} else {
  registerLog(tempfile())
}
registerLog(logfile1)
registerLog(logfile2)
```
registerLog(logcon3)

doLog("Start logging")
doLog("Do something...")
doLog("End logging")

flushLog() ## usually not needed, see notes

txt1 <- readLines(logfile1)
txt2 <- readLines(logfile2)

cat(txt1)
cat(txt2)

clearLog()

registerLog(logfile1, logfile2, logcon3)

doLog("Start logging - round 2")
doLog("Do something again...")
doLog("End logging - for good")

flushLog() ## usually not needed, see notes

txt1 <- readLines(logfile1)
txt2 <- readLines(logfile2)

cat(txt1)
cat(txt2)

## clean up files and objects to close unused connections
closeLoggerConnections()

---

**relevels**

*Relevel a factor by a named or unnamed vector.*

**Description**

This function wraps `relevelsByNamedVec` for named vector and `relevelsByNotNamedVec` for not named vectors

**Usage**

```r
relevels(
  x,
  refs,
  missingLevels = c("pass", "warning", "error"),
  unrecognisedLevels = c("warning", "pass", "error")
)
```
**Arguments**

- **x**: A factor
- **refs**: A named vector or unnamed vector.
- **missingLevels**: Actions taken in case existing levels are missing: 'pass', 'warning', or 'error'.
- **unrecognisedLevels**: Actions taken in case unrecognised levels are found: 'pass', 'warning', or 'error'.

**Value**

A vector of factor

**See Also**

`relevelsByNamedVec` and `relevelsByNotNamedVec`

**Examples**

```r
oldFactor <- factor(c("A", "B", "A", "C", "B"), levels=LETTERS[1:3])
refLevels <- c("B", "C", "A")
refDict <- c("A"="a", "B"="b", "C"="c")
newFactor <- relevels(oldFactor, refLevels)
stopifnot(identical(newFactor, factor(c("A", "B", "A", "C", "B"), levels=c("B", "C", "A")))))
newFactor2 <- relevels(oldFactor, refDict)
stopifnot(identical(newFactor2, factor(c("a", "b", "a", "c", "b"), levels=c("a", "b", "c")))))
```

---

**relevelsByNamedVec**

Relevel a factor by a named vector.

**Description**

If names contain character strings other than the levels in the old factor and warning is set to TRUE, a warning will be raised.

**Usage**

```r
relevelsByNamedVec(
  x,
  refs,
  missingLevels = c("pass", "warning", "error"),
  unrecognisedLevels = c("warning", "pass", "error")
)
```
relevelsByNotNamedVec

Arguments

- `x`: A factor
- `refs`: A named vector. The names of the vector are all or a subset of levels in the old factor. And the values are new levels
- `missingLevels`: Actions taken in case existing levels are missing: 'pass', 'warning', or 'error'.
- `unrecognisedLevels`: Actions taken in case unrecognised levels are found: 'pass', 'warning', or 'error'.

Details

The levels of the factor are the names of the ref vector, and the order of the ref vector matters: it is the levels of the new factor.

Value

A vector of factor

Examples

```r
oldFactor <- factor(c("A", "B", "A", "C", "B"), levels=LETTERS[1:3])
factorDict <- c("A"="a", "B"="b", "C"="c")
newFactor <- relevelsByNamedVec(oldFactor, factorDict)
stopifnot(identical(newFactor, factor(c("a", "b", "a", "c", "b"), levels=c("a", "b", "c"))))
## TODO: test warning and error
```

---

relevelsByNotNamedVec  Relevel a factor by a unnamed vector.

Description

If names contain character strings other than the levels in the old factor and warning is set to TRUE, a warning will be raised.

Usage

```r
relevelsByNotNamedVec(
  x,
  refs,
  missingLevels = c("pass", "warning", "error"),
  unrecognisedLevels = c("warning", "pass", "error")
)
```
reload

65

reload

Reload a package

Description

Reload a package by first detaching and loading the library.

Usage

reload(pkg)

Arguments

pkg  Character string, name of the package

Value

Side effect is used.

Note

So far only character is accepted
removeColumns

Author(s)
Jitao David Zhang <jitao_david.zhang@roche.com>

See Also
detach and library

Examples

## the example should not run because it will reload the package
## Not run:
  reload(ribiosUtils)

## End(Not run)

---

removeColumns Remove columns

Description
Remove columns from a data.frame object

Usage
removeColumns(data.frame, columns, drop = FALSE)

Arguments
data.frame data.frame
columns names of columns to be removed
drop Logical, whether the matrix should be dropped to vector if only one column is left

Details
The function is equivalent to the subsetting operation with brackets. It provides a tidy programming interface to manipulate data.frames.

Value
data.frame with specified columns removed

Author(s)
Jitao David Zhang <jitao_david.zhang@roche.com>
Examples

```r
clubs <- data.frame(Points=c(21,23,28,24), Name=c("BVB", "FCB", "HSV", "FCK"), games=c(12,11,11,12))
removeColumns(clubs,c("Name"))
```

---

**removeInvarCol**  
Remove invariable columns from a data frame or matrix

**Description**

Columns with one unique value are invariable. The functions help to remove such columns from a data frame (or matrix) in order to highlight the variables.

**Usage**

```r
removeInvarCol(df)
```

**Arguments**

- `df` A data frame or matrix

**Details**

`removeInvarCol` the data frame removing invariable column(s).

`isVarCol` and `isInvarCol` are helper functions, returning a logical vector indicating the variable and invariable columns respectively.

**Value**

`isVarCol` and `isInvarCol` return a logical vector indicating the variable and invariable columns respectively.

`removeInvarCol` removes invariable columns.

**Author(s)**

Jitao David Zhang <jitao_david.zhang@roche.com>

**Examples**

```r
testDf <- data.frame(a=1:4, b=7, c=LETTERS[1:4])
isVarCol(testDf)
isInvarCol(testDf)
removeInvarCol(testDf)
```
replaceColumnName  
*Replace column names in data.frame*

**Description**
Replace column names in data.frame

**Usage**
```r
replaceColumnName(data.frame, old.names, new.names)
```

**Arguments**
- `data.frame`: A data.frame
- `old.names`: Old column names to be replaced
- `new.names`: New column names

**Value**
Data.frame with column names updated

**Author(s)**
Jitao David Zhang <jitao_david.zhang@roche.com>

**Examples**
```r
clubs <- data.frame(Points=c(21,23,28,24), Name=c("BVB", "FCB", "HSV", "FCK"), games=c(12,11,11,12))
replaceColumnName(clubs, c("Points", "games"), c("Punkte", "Spiele"))
```

ribiosTempdir
*A temporary directory which (1) every machine in the cluster has access to and (2) has sufficient space*

**Description**
A temporary directory which (1) every machine in the cluster has access to and (2) has sufficient space

**Usage**
```r
ribiosTempdir()
```
ribiosTempfile

Value

A character string of the directory name

See Also

ribiosTempfile

Description

A temporary file which (1) every machine in the cluster has access to and (2) there is sufficient space.

Usage

ribiosTempfile(pattern = "file", tmpdir = ribiosTempdir(), fileext = "")

Arguments

pattern Character string, file pattern
tmpdir Character string, temp directory
fileext Character string, file name extension (suffix)

Value

A character string of the file name

See Also

ribiosTempdir

ribiosUtils ribiosUtils

Description

ribiosUtils is a swiss-knife package providing misc utilities

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>, with inputs from Clemens Broger, Martin Ebeling, Laura Badi and Roland Schmucki
rmat  

Remove temporary files at a specified time interval from now

Description

Send a at job to remove (probably temporary) files in the future with a specified time interval from now

Usage

rmat(..., days = NULL, hours = NULL, minutes = NULL, dry = TRUE)

Arguments

...  Files to be removed

days  Numeric, interval in days

hours  Numeric, interval in hours

minutes  Numeric, interval in minutes

dry  Logical, if set to TRUE, only the command will be returned and files are not really removed.

Details

The command will delete files, and there is usually no way to get deleted files back. So make sure you know what you are doing!

Days, hours, and minutes can be given in a mixed way: they will be summed up to give the interval.

Value

(Invisibly) the output of the at job.

Note

Since the command uses at internally, it is unlikely the command will work in the Windows system “out of box”.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

qsyste for running system commands quietly.
Examples

tmp1 <- tempfile()
tmp2 <- tempfile()
rmat(tmp1, tmp2, minutes=1)

rocheCore

Extract core identifiers from Roche compound IDs

Description

Extract core identifiers from Roche compound IDs

Usage

rocheCore(str, short = FALSE)

Arguments

str Character strings
short Logical, if TRUE, the short version of Roche identifiers (RO[0-9]{4}) is returned. Default: FALSE

Value

Core identifiers if the element is a Roche compound ID, the original element otherwise. Non-character input will be converted to character strings first.

See Also

isRocheCompoundID

Examples

rocheCore(c("R01234567-001", "R01234567-001-000", "R01234567", "R0noise-001", "anyOther-not-affected"))
rocheCore(c("R01234567-001", "R01234567-001-000", "R01234567", "R0noise-001","anyOther-not-affected"), short=TRUE)
**rowscale**  
*S3 method for row-scaling*

**Description**

S3 method for row-scaling

**Usage**

rowscale(x, center = TRUE, scale = TRUE)

**Arguments**

- `x` Any object
- `center` Logical, whether centering should be done before scaling
- `scale` Logical, whether scaling should be done

**Value**

The input object with rows scaled

---

**rowscale.matrix**  
*Scale a matrix by row*

**Description**

Scaling a matrix by row can be slightly slower due to a transposing step.

**Usage**

## S3 method for class 'matrix'

rowscale(x, center = TRUE, scale = TRUE)

**Arguments**

- `x` An matrix
- `center` Logical, passed to scale. to TRUE
- `scale` Logical, passed to scale. TRUE

**Value**

A matrix with each row scaled.

**Author(s)**

Jitao David Zhang <jitao_david.zhang@roche.com>
rsetdiff

See Also

scale

Examples

```r
mat <- matrix(rnorm(20), nrow=4)
rs.mat <- rowscale(mat)

print(mat)
print(rs.mat)
rowMeans(rs.mat)
apply(rs.mat, 1L, sd)

rowscale(mat, center=FALSE, scale=FALSE) # equal to mat
rowscale(mat, center=TRUE, scale=FALSE)
rowscale(mat, center=FALSE, scale=TRUE)
```

---

rsetdiff Reverse setdiff

Description

reverse setdiff, i.e. rsetdiff(x, y) equals setdiff(y, x)

Usage

```r
rsetdiff(x, y)
```

Arguments

- `x` a vector
- `y` another vector

Value

Similar to setdiff, but with elements in y but not in x

Author(s)

Jitao David Zhang

Examples

```r
testVec1 <- LETTERS[3:6]
testVec2 <- LETTERS[5:7]
rsetdiff(testVec1, testVec2)
```
**setupInit**

Prepare R for an interactive script

**Description**

The function prepares R for an interactive session (e.g. in a script). Currently it defines behaviour in case of errors: a file named “ribios.dump” is written.

**Usage**

`scriptInit()`

**Value**

Side effect is used.

**Author(s)**

Jitao David Zhang <jitao_david.zhang@roche.com>

**See Also**

`options`

**Examples**

```r
## do not run unless the script mode is needed
## Not run:
scriptInit()
```

**setDebug**

Functions for command-line Rscript debugging

**Description**

These functions are used to debug command-line executable Rscripts in R sessions

**Usage**

`setDebug()`
shortenStr

Details

setDebug sets the environmental variable RIBIOS_SCRIPT_DEBUG as TRUE. unsetDebug unsets the variable. isDebugging checks whether the variable is set or not. isIntDebugging tests whether the scripts runs interactively or runs in the debugging mode. The last one can be useful when debugging Rscript in a R session.

A programmer wishing to debug a Rscript can explicitly set (or unset) the RIBIOS_SCRIPT_DEBUG variable in order to activate (inactivate) certain trunks of codes. This can be automated via isDebugging, or probably more conveniently, by isIntDebugging: if the script runs in an interactive mode, or the debugging flag is set, the function returns TRUE.

Value

setDebug and unsetDebug returns an invisible value indicating whether the variable setting (unsetting) was successful.

isDebugging and isIntDebugging returns logical values.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

unsetDebug()
print(isDebugging())
setDebug()
print(isDebugging())
unsetDebug()
print(isDebugging())
print(isIntDebugging())

shortenStr

Shorten strings to a given number of characters

Description

Shorten strings to a given number of characters

Usage

shortenStr(str, nchar = 8)

Arguments

str A vector of strings
nchar The maximal number of characters to keep
Value

A vector of strings of the same length as the input, with each string shortened to the desired length. Strings with more characters than nchar will be shortened.

Note

NA will be kept as they are.

Examples

```r
inputStrs <- c("abc", "abcd", "abcde", NA)
shortenStr(inputStrs, nchar=4)
## expected outcome: abc, abcd, abcd..., NA
```

---

**silencio**

*Keep silent by suppressing warnings and messages*

Description

The function is used to keep the command silent by suppressing warnings and messages.

Usage

`silencio(...)`

Arguments

... Any function call

Value

The same as the function call

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

`suppressWarnings`, `suppressMessages`

Examples

```r
wsqrt <- function(x) { warning("Beep"); message("Calculating square"); return(x^2)}
silencio(wsqrt(3))
```
sortAndFilterByCumsumprop

Sort a numeric vector and filter by a threshold of cumsumprop

Description

Sort a numeric vector and filter by a threshold of cumsumprop

Usage

sortAndFilterByCumsumprop(x, thr = 0.9)

Arguments

x Numeric vector, usually named
thr Threshold, default 0.9, meaning that items whose proportion of cumulative sum just above 0.9 are kept.

Value

Another numeric vector, likely shorter than x, items whose cumsumprop is equal or lower than thr. The rest items are summed into one new item, with the name rest.
This function can be useful to extract from a long numeric vector the largest items that dominate the sum of the vector.

Examples

sortAndFilterByCumsumprop(x, thr=0.99) ## F and E should be returned

sortByCol

Sort data.frame rows by values in specified columns

Description

Sort rows of an data.frame by values in specified columns.

Usage

sortByCol(data.frame, columns, na.last = TRUE, decreasing = TRUE, orderAsAttr = FALSE)
Arguments

data.frame A data.frame object
columns Column name(s) which should be ordered
na.last Logical, whether NA should be sorted as last
decreasing Logical, whether the sorting should be in the decreasing order
orderAsAttr Logical, whether the order index vectors should be returned in the attribute “order” of the sorted data.frame

Details
Columns can be specified by integer indices, logical vectors or character names.

Value
Sorted data.frame

Author(s)
Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

sample.df <- data.frame(teams=c("HSV", "BVB", "FCB", "FCN"),pts=c(18,17,17,9), number=c(7,7,6,6))
sortByCol(sample.df, 1L)
sortByCol(sample.df, 1L, decreasing=FALSE)

sortByCol(sample.df, c(3L, 1L))
sortByCol(sample.df, c(3L, 1L), decreasing=FALSE)
sortByCol(sample.df, c(3L, 2L))

sortByCol(sample.df, c(TRUE, FALSE, TRUE))
sortByCol(sample.df, c("teams", "pts"))
sortByCol(sample.df, c("pts", "number", "teams"))
sortByCol(sample.df, c("pts", "teams", "number"))

sortByDimnames "Sort matrix by dim names"

Description
Rearrange rows and columns of a matrix by dim names

Usage
sortByDimnames(x, row.decreasing = FALSE, col.decreasing = FALSE)
Arguments

- `x` A matrix or data.frame
- `row.decreasing` Logical, whether rows should be sorted decreasingly
- `col.decreasing` Logical, whether columns should be sorted decreasingly

Value

Resorted matrix or data frame

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

```r
testMat <- matrix(1:16, nrow=4, dimnames=list(c("B", "D", "A", "C"), c("t", "f", "a", "g")))
sortByDimnames(testMat)
sortByDimnames(testMat, row.decreasing=TRUE, col.decreasing=FALSE)
```

strtoken

Tokenize strings by character

Description

Tokenize strings by character in a similar way as the `strsplit` function in the base package. The function can return a matrix of tokenized items when `index` is missing. If `index` is given, tokenized items in the selected position(s) are returned. See examples.

Usage

```r
strtoken(x, split, index, ...)
```

Arguments

- `x` A vector of character strings; non-character vectors are cast into characters.
- `split` A character to split the strings.
- `index` Numeric vector indicating which fields should be returned; if missing or set to `NULL`, a matrix containing all fields are returned.
- `...` Other parameters passed to `strsplit`

Value

A matrix if `index` is missing, `NULL`, or contains more than one integer indices; otherwise a character vector.
stubborngc

Description

stubborngc repeats collecting garbage until no more resource can be freed

Usage

stubborngc(verbose = FALSE, reset = TRUE)

Arguments

verbose Logical, verbose or not
reset Logical, reset or not.

Value

Side effect is used.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>
subsetByColumnName

See Also
gc

Examples

stubborngc()

subsetByColumnName(data.frame, reqCols, ignore.case = FALSE)

Arguments

data.frame A data.frame object
reqCols required columns
ignore.case logical, whether the case is considered

Value

If all required column names are present, the data.frame object will be subset to include only these columns and the result data.frame is returned. Otherwise an error message is printed.

Examples

myTestDf <- data.frame(HBV=1:3, VFB=0:2, BVB=4:6, FCB=2:4)
myFavTeams <- c("HBV", "BVB")
subsetByColumnName(myTestDf, myFavTeams)
myFavTeamsCase <- c("hbv", "bVb")
subsetByColumnName(myTestDf, myFavTeamsCase, ignore.case=TRUE)
```
summarizeRows  Summarizing rows/columns by a factor

Description
Apply a function to summarize rows/columns that assigned to the same level by a factor vector.

Usage
summarizeRows(matrix, factor, fun = mean, ...)

Arguments
matrix  A numeric matrix
factor  A vector of factors, either of the length of nrow(matrix) (for summarizeRows),
or the length of ncol(matrix) (for summarizeColumns).
fun  A function or a name for a function, the summarizing function applied to rows/columns
sharing the same level
...  Further parameters passed to the function

Details
NA levels are neglected, and corresponding rows/columns will not contribute to the summarized
matrix.
summarizeCols is synonymous to summarizeColumns

Value
A matrix, the dimension will be determined by the number of levels of the factor vector.

Author(s)
Jitao David Zhang <jitao_david.zhang@roche.com>

Examples
my.matrix <- matrix(1:25, nrow=5)
print(my.matrix)
my.factor <- factor(c("A", "B", "A", "C", "B"))
summarizeRows(matrix=my.matrix, factor=my.factor, fun=mean)
summarizeRows(matrix=my.matrix, factor=my.factor, fun=prod)
summarizeColumns(matrix=my.matrix, factor=my.factor, fun=mean)
summarizeColumns(matrix=my.matrix, factor=my.factor, fun=prod)

## NA values in factor
```
my.na.factor <- factor(c("A", "B", "A", "C", NA))
summarizeRows(matrix=my.matrix, factor=my.na.factor, fun=mean)
summarizeRows(matrix=my.matrix, factor=my.na.factor, fun=prod)
summarizeColumns(matrix=my.matrix, factor=my.na.factor, fun=mean)
summarizeColumns(matrix=my.matrix, factor=my.na.factor, fun=prod)

---

**trim**

*Trim leading and tailing spaces from string*

**Description**

The function trims leading and/or tailing spaces from string(s), using C function implemented in the BIOS library.

**Usage**

```r
trim(x, left = " \
\r\t", right = " \
\r\t")
```

**Arguments**

- `x` A character string, or a vector of strings
- `left` Characters that are trimmed from the left side.
- `right` Characters that are trimmed from the right side

**Details**

`left` and `right` can be set to NULL. In such cases no trimming will be performed.

**Value**

Trimmed string(s)

**Author(s)**

Jitao David Zhang <jitao_david.zhang@roche.com>

**Examples**

```r
myStrings <- c("This is a fine day\n",
               " Hallo Professor!",
               " NUR DER HSV ")
trim(myStrings)
```
uniqueLength  \hspace{1cm}  \textit{Length of unique elements in a vector}

\textbf{Description}

Length of unique elements in a vector

\textbf{Usage}

\texttt{uniqueLength(x, incomparables = FALSE)}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{x} \hspace{1cm} A vector
  \item \texttt{incomparables} \hspace{0.5cm} See \texttt{unique}
\end{itemize}

\textbf{Value}

An integer indicating the number of unique elements in the input vector

\textbf{Author(s)}

Jitao David Zhang <jitao_david.zhang@roche.com>

\textbf{See Also}

\texttt{unique}

\textbf{Examples}

\begin{verbatim}
test.vec1 <- c("HSV", "FCB", "BVB", "HSV", "BVB")
uniqueLength(test.vec1)

test.vec2 <- c(1L, 2L, 3L, 5L, 2L, 1L, 5L)
ulen(test.vec2)
\end{verbatim}
uniqueNonNA

Make a vector free of NA and unique

Description
Make a vector free of NA and unique

Usage
uniqueNonNA(x)

Arguments
x A vector

Value
A unique vector without NA

Examples

testVec <- c(3,4,5,NA,3,5)
uniqueNonNA(testVec)

verbose
Print messages conditional on the verbose level

Description
The verbose level can be represented by non-negative integers. The larger the number is, the more
verbose is the program: it prints then more messages for users’ information.

Usage
verbose(..., global = 1L, this = 1L)

Arguments
... Messages to be printed, will be passed to the message function
global Integer, the global verbose level
this Integer, the verbose level of this message
Details

This function decides whether or not to print a message, dependent on the global verbose level and the specific level of the message. If the specific level is larger than the global level, the message is suppressed; otherwise it is printed. See the details section for an example.

Suppose the global verbose level is set to 5, and two messages have levels of 1 and 7 respectively. Since 1 suggests a low-threshold of being verbose, the first message is printed; whereas the message of level 7 is only printed when the program should run in a more verbose way (7, 8, 9, ..., 9), it is suppressed in the current global verbose level.

Value

The function is used for its side effect by printing messages.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

Gv <- 5L
verbose("Slightly verbosing", global=Gv, this=1L)
verbose("Moderately verbosing", global=Gv, this=5L)
verbose("Heavily verbosing", global=Gv, this=9L)

whoami

System user name

Description

System user name

Usage

whoami()

Value

System user name

Examples

whoami()
The function `writeLog` can be used to log outputs and/or running status of scripts to one connection. To use it one does not need to run `registerLog` first.

### Usage

`writeLog(fmt, ..., con = stdout(), level = 0)`

### Arguments

- `fmt` Format string to passed on to `sprintf`
- `...` Parameters passed on to `sprintf`
- `con` A connection, for instance a file (or its name) or `stdout()`
- `level` Logging level: each higher level will add one extra space before the message. See examples

### Details

In contrast, `doLog` can be used to log on multiple connections that are registered by `registerLog`. Therefore, to register logger(s) with `registerLog` is a prerequisite of calling `doLog`. Internally `doLog` calls `writeLog` sequentially to make multiple-connection logging.

### Value

Side effect is used.

### Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

### See Also

- `registerLog` to register more than one loggers so that `doLog` can write to them sequentially.

### Examples

```r
writeLog("This is the start of a log")
writeLog("Message 1", level=1)
writeLog("Message 1.1", level=2)
writeLog("Message 1.2", level=2)
writeLog("Message 2", level=1)
writeLog("Message 3", level=1)
writeLog("Message 3 (special)", level=4)
```
writeLog("End of the log");

## log with format
writeLog("This is Message %d", 1)
writeLog("Square of 2 is %2.2f", sqrt(2))

## NA is handled automatically
writeLog("This is a not available value: %s", NA, level=1)
writeLog("This is a NULL value: %s", NULL, level=1)
Index

allIdentical, 4
apply1 (summarizeRows), 82
apply2 (summarizeRows), 82
asNumMatrix, 5
assertColumnName, 5, 81
assertContrast, 6, 8
assertDesign, 7, 8
assertDesignContrast, 7
assertDir, 10
assertDir (isDir), 28
assertFile, 29
assertFile (checkFile), 9
atoMatrix (asNumMatrix), 5
basefilename (extname), 20
bedaInfo, 8, 54
biomicsPstorePath2URL, 9
bmp, 49
checkDir (isDir), 28
checkFile, 9, 29
chosenFew, 10
clearLog (registerLog), 60
closeFileDevice (openFileDevice), 48
closeLoggerConnections, 11
columnOverlapCoefficient, 12
compTwoVecs, 12
countTokens, 13
createDir, 14
cumJaccardDistance (cumJaccardIndex), 15
cumJaccardIndex, 15
cumOverlapCoefficient, 15, 16
cumOverlapDistance
cumsumprop, 17
cut, 18
cutInterval, 17
detach, 66
dfFactor, 19
dfFactor2Str, 20
dir.create, 14
doLog (writeLog), 87
dir (isDir), 28
distMat, 29
directOverlapCoefficient, 20
extname, 20, 49
file.info, 29
fixWidthStr, 21
flushLog (registerLog), 60
gc, 81
ggetDefaultFontFamily, 22
haltifnot, 23
head, 24, 25
headhead, 24
headtail, 25
identical, 4, 43
identicalMatrix, 26
identicalMatrixValue, 26
imatch, 27
imatchv (imatch), 27
intersect, 45
ipmatch (imatch), 27
ipmatchv (imatch), 27
isDebugging (setDebug), 74
isDir, 10, 28
isError, 29
isIntDebugging (setDebug), 74
isInvarCol (removeInvarCol), 67
isMaxStatRow (keepMaxStatRow), 31
isRocheCompoundID, 30, 71
isVarCol (removeInvarCol), 67
jaccardDistance (jaccardIndex), 31
jaccardIndex, 31, 50
jpeg, 49
keepMaxStatRow, 31
keepMaxStatRowInd (keepMaxStatRow), 31
libordie, 33
library, 66
list2df, 35
listOverlapCoefficient, 35
long2df2matrix, 36
match, 28, 39
matchColumn, 37
matchColumnIndex (matchColumn), 37
matchColumnName, 5, 39
matchv (imatch), 27
matrix2longdf, 40
mergeInfreqLevelsByCumsumprop, 41
midentical, 42
mintersect (munion), 45
mmatch, 43
mset (munion), 45
msetdiff (munion), 45
munion, 45
na.false, 46
naivePairwiseDist, 46
nField (countTokens), 13
ofactor, 47
openIODevice, 48
options, 74
overlapCoefficient, 49
overlapDistance (overlapCoefficient), 49
pAbsLog10Score, 50, 54, 55
pairwiseJaccardDistance
  (pairwiseJaccardIndex), 51
pairwiseJaccardIndex, 51
pairwiseOverlapCoefficient
  (pairwiseOverlapDistance), 52
pairwiseOverlapDistance, 52
pdf, 49
percentage, 53
pmatch, 28
png, 49
pQnormScore, 51, 53, 55
print.BEDAinfo, 54
pScore, 55
putColsFirst, 56
pwdecode, 56
pwencode, 57
qqmsg, 34, 58
qsystem, 59, 70
quit, 58
refactorNum, 59
registerLog, 11, 60
relevels, 62
relevelsByNamedVec, 62, 63, 64
relevelsByNotNamedVec, 62, 63, 64
reload, 65
removeColumns, 66
removeInvarCol, 67
replaceColumnName, 68
ribiosTempdir, 68, 69
ribiosTempfile, 69, 69
ribiosUtils, 69
rmat, 70
rocheCore, 71
rowscale, 72
rowscale.matrix, 72
rsetdiff, 73
scale, 73
scriptInit, 74
setDebug, 74
setdiff, 45
shortenStr, 22, 75
silencio, 76
sortAndFilterByCumsumprop, 77
sortByCol, 77
sortByDimnames, 78
stop, 23
stopifnot, 23
stringDataFrame2numericMatrix
  (asNumMatrix), 5
strsplit, 14, 79, 80
strtoken, 14, 79
stubbornGc, 80
subsetByColumnName, 81
summarizeCols (summarizeRows), 82
summarizeColumns (summarizeRows), 82
summarizeRows, 82
suppressMessages, 76
suppressWarnings, 76
tail, 24, 25
tailtail (headhead), 24
tiff, 49
trim, 83
ulen (uniqueLength), 84
union, 45
unique, 84
uniqueLength, 84
uniqueNonNA, 85
unsetDebug (setDebug), 74

verbose, 85

warning, 23
whoami, 86
writeLog, 87