Package 'RQDA'

February 19, 2015

Type Package

Title R-based Qualitative Data Analysis

Version 0.2-7

Date 2014-10-27

Author HUANG Ronggui

Maintainer HUANG Ronggui <ronggui.huang@gmail.com>

Depends R (>= 2.8.0), RSQLite(>= 1.0.0), gWidgetsRGtk2 (>= 0.0-36)

Imports RGtk2 (>= 2.20), DBI, igraph, gWidgets (>= 0.0-31)

Enhances tcltk, rjpod, d3Network

Description

R package for Qualitative Data Analysis. Current version only supports plain text, but it can import PDF highlights if the Enhance package of rjpod, which is available on R-Forge, is installed.

Additional_repositories http://r-forge.r-project.org/

License BSD_3_clause + file LICENSE

LazyLoad yes

SystemRequirements Java (>= 1.6) for rjpod

URL http://rqda.r-forge.r-project.org/

NeedsCompilation no

Repository CRAN

Date/Publication 2014-10-27 20:19:04

R topics documented:

RQDA-package 2
casesCodedBy 3
codingBySearch 4
crossCodes 5
Deletion 7
exportCodedFile 8
exportCodings 9

filesByCodes . . . 10
filesCodedBy . . . 10
getAttr . . . 11
getCases . . . 13
getCodingsByOne . . . 14
getCodingTable . . . 15
getFileIdSets . . . 16
getFiles . . . 17
getMemos . . . 18
gselect.list . . . 19
nCodedByTwo . . . 20
Ops.codings . . . 20
Ops.RQDA . . . 22
Project . . . 23
queryFiles . . . 24
relation . . . 25
retrieval . . . 26
RQDAQuery . . . 27
RQDATables . . . 28
searchFiles . . . 32
summaryCodings . . . 34
viewPlainFile . . . 35
write.FileList . . . 36

Index **37**

`RQDA-package` *R-based Qualitative Data Analysis package*

Description

Qualitative Data Analysis based on R language. Current version supports plain text. In addition, it can import PDF highlights.

Details

The workhorse function for end-user is the RQDA(), you can use RQDA() to start the GUI after library(QRDA). Please Refer to the documentation section of the project homepage for the usage of RQDA.

The position of ViewFile widget can be controlled by "widgetCoordinate" options, with default value c(400,2). You can change it by options("widgetCoordinate"=c(x,y)), where x and y is integer specifying the position.

The size of many widgets (e.g. ViewFile widgets) can be controlled by "widgetSize" options, with default value c(550,700). You can change it by options("widgetSize"=c(x,y)), where x and y is integer specifying width and height.

Author(s)

Huang Ronggui

Maintainer: Huang <ronggui.huang@gmail.com>

References

Kelle, U. (ed). 1995. "Computer-aided qualitative data analysis: theory, methods and practice." Sage Publications.

Lewins, A. & Silver, C.2007. Using Software in Qualitative Research : A Step-by-Step Guide. Sage Publications.

Examples

```
## Not run: library(RQDA)
RQDA()

## End(Not run)
```

`casesCodedBy` *Get cases coded by specific codes, by specifying the code IDs.*

Description

Get cases coded by specific codes, by specifying the code IDs.

Usage

```
casesCodedByAnd(cid)

casesCodedByOr(cid)

casesCodedByNot(cid)
```

Arguments

`cid`	an integer vector of code IDs.

Details

`casesCodedByAnd` returns case IDs which are coded by all codes from cid.

`casesCodedByOr` returns case IDs which are coded by any code from cid.

`casesCodedByNot` returns case IDs which are not coded by any code from cid.

Value

a vector of file IDs, with class of "RQDA.vector" and "caseId".

Author(s)

Ronggui HUANG

See Also

getCaseNames

Examples

```
## Not run:
filesCodedByAnd(1:2) ## coded by code 1 and 2

filesCodedByOr(1:2) ## coded by code 1 or 2

filesCodedByNot(1:2) ## coded by codes other than 1 and 2

## End(Not run)
```

---

codingBySearch *Auto-coding by pattern matching*

Description

Applies the specified code to a specified file based on given text pattern.

Usage

```
codingBySearch(pattern, fid = getFileIds(), cid, seperator="\n",
               concatenate = FALSE, ...)
```

Arguments

pattern	a text string, to be matched to the text in the RQDA project file
fid	File id, in the GUI 'File' tab click on file to see its id
cid	Code id, in the GUI 'Codes' tab click on code to see its id
seperator	single character string, specifying the seperator of unit of analysis.
concatenate	a boolean value, if TRUE then matches in adjacent units (ie only separated by 'seperator') are fused into a single coding.
...	arguments passed to gregexpr. A useful example is the 'ignore.case' argument.

Details

The function can be used if on is interested in automatically applying a code to paragraphs in a file based on certain words specified by 'pattern'.

It first splits the whole text into pieces which depends on the seperator, then match the pattern with each pieces. When a matche is found, tha piece is coded to the code specified by cid. The default seperator defines paragraph as unit of analysis. The seperator is passed to the pattern argument of gregexpr.

This function is also useful for keyword in context (KWIC) analysis.

Value

The function is used for its side effect.

Author(s)

Ronggui Huang

See Also

getFileIds

Examples

```
## Not run:
codingBySearch("internet",fid=1,cid=2)

codingBySearch("internet",fid=getFileIds(),cid=2)

codingBySearch("internet",fid=getFileIds(),cid=2, seperator="[.!?]")

## End(Not run)
```

---

crossCodes *Inter-codes relationship*

Description

Return a matrix, give a summary of inter-codes relationship.

Usage

```
crossCodes(relation=c("overlap","inclusion","exact","proximity"),codeList=NULL,
          data=GetCodingTable(),print=TRUE,...)

crossTwoCodes(cid1,cid2,data,relation=c("overlap","inclusion","exact","proximity"),...)
```

Arguments

relation	The relation between codes
codeList	A character vector, the codes list on which the inter-code relationship is based
data	Data frame return by getCodingTable,may be subset of the full coding table
print	When TRUE, print the results automatically
cid1	Length-1 code id. It is numeric.
cid2	Length-1 code id. It is numeric.
...	... is not used yet.

Details

The inter-codes relationship calculation is based on the relationship between the associated codings of the codes.

Giving the code name list (a character list), crossCodes returns the inter-relationship of 2 or more than 2 codes. crossCodes make heavy use of for loops, so it may takes a while to get the result when the coding table is large.

crossTwoCodes returns the summary of inter-codes relationship of two codes based on the code id (each code id is a length-1 integer vector).

Value

For crossCodes, it is a matrix. The upper matrix contains the number of codings fitting the relation between the respective two codes. the lower matrix is all NA. rownames of the matrix is the name of the codes , and the colnames of the matrix is the corresponding id of codes.

For crossCodes, it is a numeric vector.

See Also

relation

Examples

```
## Not run:
crossCodes()

## End(Not run)
```

Deletion *Functions for dealing with the temporarily deleted data*

Description

`list.deleted` shows the temporarily deleted data (deleted by delete button, which is only tagged with deletion mark in the *.rqda file). `pdelete` permanently deletes them. `CleanProject` cleans the *.rqda file (call pdelete with every possible value for the type argument). `undelete` removes the temporarily deletion mark to reuse the temporarily deleted data.

Usage

```
list.deleted(type=c("file","code","case","codecategory","filecategory"))

pdelete(type=c("file","code","case","codecategory","filecategory","coding"),
        ask=FALSE)

CleanProject(ask=FALSE)

undelete(type=c("file","code","case","codecategory","filecategory"),ask=TRUE)
```

Arguments

`type`	Types of elements in the *.rqda file. "file" is the name of file (in the Files Tab). "code" is the name of codes (in the Codes Tab). "case" is the name of case (in the Case Tab). "codecategory" is name of code category (in the C-Cat Tab). "filecategory" is name of file category (in the F-Cat Tab). "coding" is the text segment associated with specific code.
`ask`	You can choose which ones to be deleted when is TRUE. Otherwise, it will delete all with temporarily deletion mark.

Details

By GUI, you can delete file, code, case, code category and file categroy. When click the delete button, the status of related elements (e.g. for file, the elements includ file, related coding, related case category and file category) are set from 1 to 0. In this sense, deletion from GUI is temporary. After that, you can use `list.deleted` to show which ones are tagged as deleted. By `pdelete`, you can permenantly delete those tagged with temporarily deletion mark. By `undelete`, you can undo the temporary deletion, the status of related elements are set back to 1.

When ask is FALSE, it will apply to all the propriate elements of specific type. When it is TRUE, you can choose the elements of the specific type which the action (pdelet or undelete) applies to.

Value

For `list.deleted`, a data frame if there are some records with temporarily deletion mark for the specified type. For `pdelete`, `CleanProject` and `undelete`, no value is return. These functions are used for the side-effects.

Note

In order to make the temporarily deletion of code and the associated coding can be undeleted again, RQDA differentiates the temporarily deletion of codings (which are deleted by deleting a code) from that produced by unmark button in the Coding Tab: the former with status = 0 while the latter with status = -1.

Author(s)

Ronggui HUANG

---

exportCodedFile *exported a coded file to HTML fiel*

Description

Exported a coded file to a HTML file with codings and code labels

Usage

```
exportCodedFile(file, fid, closeAfter = TRUE)
```

Arguments

file	character string to specify the HTML file path.
fid	the file id for export.
closeAfter	When TRUE, the file widget is closed after export.

Value

No returned value, for its side effect only.

Author(s)

Ronggui HUANG

Examples

```
## Not run:
exportCodedFile("~/coded file with id of 1.html", fid=1)

## End(Not run)
```

`exportCodings` *Export codings to a HTML file.*

Description

To export retrieved codings to a HTML file.

Usage

```
exportCodings(file = "Exported Codings.html", Fid = NULL,
              order = c("fname", "ftime", "ctime"), append = FALSE,
              codingTable="coding")
```

Arguments

`file`	Length-one character vector, specify the name of exported file.
`Fid`	Integer vector of file id.The retrieved codings are from this subset of files. When is NULL, it means all the files.
`order`	Specify the order of retrieved codings. see details for the meanings.
`append`	Logical, when TRUE the exported codings are appended to the existing file (if exists); otherwise, it overwrites the existing file.
`codingTable`	name of sqlite data table where codings are stored. It should be either "coding" or "coding2"

Details

"fname" means order the codings by file names, "ftime" by file imported time, and "ctime" by time of coding.

Value

A html file.

Author(s)

HUANG Ronggui

filesByCodes *Relation between files and codes.*

Description

Return a data frame which indicates what codes are associted with each file.

Usage

```
filesByCodes(codingTable = c("coding", "coding2"))
```

Arguments

codingTable	name of the coding table in rqda database.

Details

The result is a data frame. Each row represents one file, and each variable represents one code. If a file is coded by a code, the value of that variable is 1, otherwise is 0.

Value

A data frame.

Author(s)

Ronggui HUANG

filesCodedBy *Get files coded by specific codes, by specifying the code IDs.*

Description

Files coded by a specific set of codes.

Usage

```
filesCodedByAnd(cid,codingTable=c("coding","coding2"))

filesCodedByOr(cid,codingTable=c("coding","coding2"))

filesCodedByNot(cid,codingTable=c("coding","coding2"))
```

Arguments

cid	an integer vector of code IDs.
codingTable	name of coding table.

Details

`filesCodedByAnd` returns file IDs which are coded by all codes from cid.

`filesCodedByOr` returns file IDs which are coded by any code from cid.

`filesCodedByNot` returns file IDs which are not coded by any code from cid.

Value

a vector of file IDs, with class of "RQDA.vector" and "fileId".

Author(s)

Ronggui HUANG

See Also

`getFileNames`

Examples

```
## Not run:
filesCodedByAnd(1:2) ## coded by code 1 and 2

filesCodedByOr(1:2) ## coded by code 1 or 2

filesCodedByNot(1:2) ## coded by codes other than 1 and 2

## End(Not run)
```

---

getAttr *attributes*

Description

Get the attributes of case or file.

Usage

```
getAttr(type = c("case", "file"),attrs = svalue(.rqda$.AttrNamesWidget),
        subset)

showSubset(x,...)
```

Arguments

type	Type of attributes.
attrs	character vector, subset of attributes to retrieve.
subset	when subset is not missing, return subset only.
x	an object from getAttr
...	Not used currently.

Details

You can add and modify the attributes of cases or files. getAttr returns this attributes as a data frame.

Sometimes, you only want to show a subset of files or cases according to their attributes. You can do the subset operation of the result from getAttr and pass it to showSubset, or you can pass a subset argument to GetAttr. The meaning of subset is the same as that in subset function.

Value

For getAttr, when type is "case", it is a data frame with class of "CaseAttr"; when type is "file", it is a data frame with class of "FileAttr". For showSubset, no value is returned, the side-effect is to change the file list or case list in respective widget.

Note

All the variables in the data frame is of class "character", you need to convert to suitable class when conducting statistical analysis.

Author(s)

HUANG Ronggui

Examples

```
## Not run:
attr <- getAttr("case")
showSubset(subset(attr,attribute1==1)) ## assuming there is a variable
named atttribute1 in attr.

## End(Not run)
```

getCases *Get the Case ID and Case Name.*

Description

Return cases IDs or names which a set of files belong to.

Usage

```
getCaseIds(fid = GetFileId(), nFiles = FALSE)

getCaseNames(caseId = GetCaseId(nFiles = FALSE))

getCases(fid, names = TRUE)
```

Arguments

fid	numeric vector, the file IDs.
nFiles	logical, return the number of files that belong to a case.
caseId	numeric vector, the case IDs.
names	logical.

Details

GetCaseId returns the case IDs which a file belongs to given the file IDs.

GetCaseName returns the case Names given the case IDs.

getCases returns the case Names or IDs depending on the argument of names. It is a wrapper of GetCaseId and GetCaseName.

Value

GetCaseId returns a data frame of two columns when nFiles is TRUE, and a numeric vector when FALSE.

GetCaseName returns a character vector or NULL if no cases are associated with the file IDs.

getNames return the names of cases when names is TRUE, id of files when FALSE.

Author(s)

HUANG Ronggui

See Also

See Also getFileIds

Examples

```
## Not run:
GetCaseName(GetCaseId(GetFileId("filecategory")))

## End(Not run)
```

`getCodingsByOne` *Return codings of one code.*

Description

get codings of a code.

Usage

```
getCodingsByOne(cid, fid=NULL,codingTable=c("coding","coding2"))
```

Arguments

`cid`	code id, an integer.
`fid`	file id, an integer vector.
`codingTable`	name of coding table.

Details

It gets codings of a code with cid from files which are specified by fid.

Value

a data frame with additional class of "codingsByOne".

Author(s)

Ronggui HUANG

See Also

`%and%`, `%or%`, `%not%`

Examples

```
## Not run:
getCodingsByOne(1)

## End(Not run)
```

getCodingTable *Get the information of codings*

Description

Get the information of codings.

Usage

```
getCodingTable()
```

Details

Codings are stored in the coding table of *.rqda file. The coding table contains necessary information, but not informative to end-users. For example, it has id of code list and file list, but not the name of them, which are stored in freecode table and source table respectively. `GetCodingTable` joins information from the three tables, and returns more informative data. See value section on the the returned components.

Value

A data frame:

`cid`	Code id
`fid`	File id
`codename`	Code name, in accordance with cid
`filename`	File name, in accordance with fid
`CodingLength`	The number of characters in the coding
`index1`	beginning index of a coding
`index2`	end index of a coding

Author(s)

HUANG Ronggui

`getFileIdSets` *Get file id from sets.*

Description

Get the file id from file-sets given the type of relation between sets. File-set is defined by the case or filecategory.

Usage

```
getFileIdSets(set = c("case", "filecategory"), relation = c("union", "intersect"))
```

Arguments

`set`	type of set, either "case" or "filecategory".
`relation`	relation between sets. either "union" or "interset".

Details

File-set is deined by case or file category. files belonging to a case/filecategory are in a set. This function get file id from the selected sets. When multiple sets are selected, the relation between them can be define. When relation is union, file ids from either selected set are returned. When reltion is intersect, only file ids appear in all the selected sets are returned.

Value

A numeric vector or NULL if no file id is well-defined.

Author(s)

HUANG Ronggui

See Also

`retrieval`, `getFileIds`

getFiles *Get the ids or names of files list*

Description

Get the ids or names of files list.

Usage

```
getFileIds(condition = c("unconditional", "case", "filecategory","both"),
          type = c("all", "coded", "uncoded","selected"))

getFileNames(fid = GetFileId())

getFiles(condition = c("unconditional", "case", "filecategory", "both"),
         type = c("all", "coded", "uncoded", "selected"), names = TRUE)
```

Arguments

condition	Any one of "unconditional", "case", "filecategory" or "both".
type	Any one of "all", "coded" or "uncoded","selected".
fid	integer vector, the id of files.
names	logical.

Details

The imported files are stored in a data base table (called source) in the *.rqda file. Every file in the source table has a unique id. Besides, every file can be assigned to a case or file category.

Given that files meet the condition, the type argument "all" means all files, "coded" means the coded files, "uncoded" means the uncoded files and "selected" means the selected files; in "files" widget, "files of case" widget and "files of category" widget respectively.

When condition is "both", the result is intersection of File Id of "case" and "filecategory".

GetFileId returns the id of files which fit the combined criterion of condition and type.

Value

Normally, it is a numeric vector of file id. If condition is "case" or "filecategory" but no case or file category is selected, it retuns NULL.

getFiles returns a vector of file IDs (with class of "RQDA.vector" and "fileId") when names is FALSE, and a vector of file names ((with class of "RQDA.vector" and "fileName") when names is TRUE.

Author(s)

HUANG Ronggui

See Also

retrieval, getFileIdSets

Examples

```
## Not run:
GetFileId() ## Id of all files
GetFileId("unconditional","coded") ## id of all coded files.
GetFileId("case","uncoded") ## id of uncoded files for the selected case.
GetFileId("filecategory","all") ## id of all files in the selected file category.

## End(Not run)
```

---

getMemos *Collection of code memos*

Description

This function collects all code memos into an object and displays them in a widget.

Usage

```
getMemos(type = "codes")
```

Arguments

type	Currently, only "codes" is supported.

Value

An object of class c("memos", "Info4Widget", "data.frame").

Author(s)

Ronggui HUANG

gselect.list *Select Items from a List*

Description

Select item(s) from a character vector.

Usage

```
gselect.list(list, multiple = TRUE, title = NULL, width = 200, height = 500, ...)
```

Arguments

list	character vector. A list of items.
multiple	logical: can more than one item be selected?
title	optional character string for window title.
width	integer. width of the widget.
height	integer. heighth of the widget.
...	Not used currently.

Details

GTK version of select.list.

Value

A character vector of selected items with encoding of UTF-8. If no item was selected and click 'OK', it returns length 0 character vector. If click 'Cancel', '""' is returned.

Note

The license of this function is subject to interpretation of the first author.

Author(s)

John Verzani and Ronggui HUANG

See Also

select.list

Examples

```
## Not run:
select.list(sort(.packages(all.available = TRUE)))

## End(Not run)
```

nCodedByTwo *Show the relationship between *codedBy using a matrix.*

Description

It returns the number of files or cases coded by two code in a matrix form.

Usage

```
nCodedByTwo(FUN, codeList = NULL, print = TRUE, ...)
```

Arguments

FUN	a function. It is usually a function from filesCodeBy* and casesCodedBy*.
codeList	character vector of code names.
print	logical, print the result automatically when TRUE.
...	not used currently.

Author(s)

Ronggui HUANG

Ops.codings *Boolean operation on codings.*

Description

Return the result codings of the Boolean operation.

Usage

```
and(CT1, CT2)

or(CT1, CT2)

not(CT1, CT2)
```

Arguments

CT1	Coding of code one.
CT2	Coding of code two.

Details

CT1 and CT2 are subset of `getCodingTable` of a specific code or returned value of `getCodingsByOne`. In former situation, only columns of "index1", "index2","fid","filename" from CT1 and CT2 are used by this function.

These functions are the same as `%and%`, `%or%`, `%not%`.

Value

An object of class "codingsByOne" and "data.frame". It consists:

`index1`

`index2`

`fid`

`filename`

`rowid`

`coding` The codings, or the text segements.

Author(s)

Ronggui HUANG

See Also

`relation`, `getCodingTable`, `%and%`

Examples

```
## Not run:
a <- getCodingTable()
c1 <- subset(a,cid==6)
c2 <- subset(a,cid==24)
ans <- and(c1, c2)
ans ## put it into a widget for inspection

## another way to do the same
and(getCodingsByOne(6), getCodingsByOne(24))

## or operator
or(getCodingsByOne(6), getCodingsByOne(24))

## not operator
not(getCodingsByOne(6), getCodingsByOne(24))

## End(Not run)
```

`Ops.RQDA` *Binary operations of some types of RQDA objects*

Description

Binary operations of RQDA.vector or codingsByOne.

Usage

```
e1 %and% e2

e1 %or% e2

e1 %not% e2
```

Arguments

`e1`	a RQDA object.
`e2`	a RQDA object.

Details

e1 and e2 can be objects of class "RQDA.vector" includes classes of "fileId", "fileName", "caseId", "caseName". They can be objects of class "codingsByOne", see `getCodingsByOne`. e1 and e2 must be the same class.

For class of "RQDA.vector", `%and%` is the `intersect` of e1 and e2. `%or%` is the `union` of e1 and e2. `%not%` is the defined as `setdiff(e1, e2)`.

Value

an object with the same structure and class of e1 and e2.

Author(s)

HUANG Ronggui

See Also

`intersect`, `union`, `setdiff`

Examples

```
## Not run:
filesCodeByAnd(1:2) %and% filesCodeByAnd(3) ## files coded by 1 and 2 as well as 3
filesCodeByAnd(1:2) %or% filesCodeByAnd(3) ## files coded by 1 and 2 or 3
filesCodeByAnd(1:2) %not% filesCodeByAnd(3) ## files coded by 1 and 2 but not 3

getCodingsByOne(1) %or% getCodingsByOne(2) ## codings of 1 or 2.
```

```
## End(Not run)
```

`Project` *Open and close project.*

Description

To open or close a project (a *.rqda file) by command.

Usage

```
openProject(path, updateGUI = FALSE)

closeProject(conName = "qdacon", assignenv = .rqda, ...)
```

Arguments

`path`	The path of of the *.rqda project file.
`updateGUI`	When TRUE, also update information on the GUI widgets.
`conName`	Do not change it.
`assignenv`	Do not change it.
`...`	Do not change it.

Details

These functions corresponde the internal functions of the "open project" and "close project" buttons.

Value

No value is returned. For the side-effect only.

Author(s)

Ronggui HUANG

queryFiles *Retrieval of file names according to their codings.*

Description

To retrieve file names according to their codings.

Usage

```
queryFiles(or=NULL, and = NULL, not = NULL, names = TRUE)
```

Arguments

or	integer vector of code id.
and	integer vector of code id.
not	integer vector of code id.
names	logical, returns file names when TRUE.

Details

Let fid.or are files coded by any code from or, fid.and are files coded by all codes of and, and fid.not are files not coded by any code of not. Then the result is `setdiff(intersect(fid.or, fid.and), fid.not)`.

This function is succeeded by `filesCodedByAnd`, `filesCodedByOr`, `filesCodedByNot` and their operators.

Value

A vector of file id when names is FALSE. A vector of file names, with the side effect of updating files widget with these file names when names is TRUE.

Author(s)

HUANG Ronggui

Examples

```
## Not run:
QueryFile(or=1:2) ## files coded to code 1 or 2.
QueryFile(and=1:2) ## files coded to code 1 and 2.
QueryFile(or=1:2, not=3:4) ## files coded to code 1 or 2 but neither 3 nor 4.

## End(Not run)
```

`relation` *Relation between two codings*

Description

To calculate the relation between two codings, given the coding indexes.

Usage

```
relation(index1, index2)
```

Arguments

`index1`	The first coding index, it is length-2 integer vector with the first element (index1[1]) less than the second element (index1[2]).
`index2`	The second coding index, it is length-2 integer vector with the first element (index2[1]) less than the second element (index2[2]).

Details

The relation between two codings can be any of inclusion, overlap, exact (special case of inclusion and overlap) and proximity (Neither overlap nor inclusion). It should be noted that two adjacent codings are regarded as proximity with distance of 0.

Value

A 6-element list:

`Relation`	Length-1 character, standing for the type of relation. It may be one of inclusion, overlap, exact or proximity.
`OverlapIndex`	Length-2 vector, the index of overlapping between two coding indexes. It is c(NA,NA) when relation is proximity.
`UnionIndex`	Length-2 vector, the index of union of the two coding indexes. It is c(NA,NA) when relation is proximity.
`Distance`	Distance of two coding indexes. It is NA when relation is not proximity.
`WhichMin`	Which argument (index1 or index2) has the minimum value. If both have the same minmum value, return NA.
`WhichMax`	Which argument (index1 or index2) has the maximum value. If both have the same maxmum value, return NA.

Author(s)

HUANG Ronggui

Examples

```
## Not run:
relation(c(20,30),c(22,28)) # inclusion
relation(c(10,40),c(20,80)) # overlap
relation(c(10,20),c(30,50)) # proximity with distance of 10
relation(c(10,20),c(20,50)) # proximity with distance of 0
relation(c(10,20),c(10,20)) # exact
relation(c(10,20),c(10,30)) # WhichMin is c(1,2)

## End(Not run)
```

`retrieval` *Retrieval of codings conditional on the file id.*

Description

To retrieve the codings of a selected code from specific set of files.

Usage

```
retrieval(Fid = NULL, order = c("fname", "ftime", "ctime"),
          CodeNameWidget = .rqda$.codes_rqda,codingTable="coding")
```

Arguments

`Fid`	Numeric vector, the file id.
`order`	The method of sort of retrieved codings.
`CodeNameWidget`	The name of code list widget.
`codingTable`	name of sqlite data table where codings are stored. It should be either "coding" or "coding2"

Details

This function retrievs the codings of a selected code from `CodeNameWidget`, given that all the codings are from a set of files which are determined by `Fid`.

Value

A `gtext` widget is open and all the codings are pushed into that widget.

Author(s)

HUANG Ronggui

See Also

`getFileIds`

RQDAQuery *Execute a SQL statement on the open *.rqda file.*

Description

Submits and executes an arbitrary SQL statement on the open *.rqda file.

Usage

```
RQDAQuery(sql)
```

Arguments

sql	a character vector of length 1 with the SQL statement.

Details

It is a wrapped version of query, to make it more convenient to submit and execute a SQL statement.

Value

The same of query, possible NULL (for the side effects of sql on the *.rqda file) or a data.frame with the output (if any) of the query.

Author(s)

HUANG Ronggui

See Also

See Also as query

Examples

```
## Not run:
RQDAQuery("select name from source where status=1")

## End(Not run)
```

RQDATables *Data Tables in rqda file*

Description

The internal data table structures in rqda file, which is a SQLite data base.

Details

Table "annotation" contatins file annotations.

fid:	file id.
position:	position of annotation.
annotation:	content of annotation.
owner:	owner of annotation.
date:	created date.
dateM:	not used currently.
status:	1 for standard status and 0 for temporarily deleted annotation.

Table "attributes" contatins information about the name list of attributes. They are held in the widget of ".AttrNamesWidget".

name:	name of attributes.
status:	1 for standard status and 0 for a temporarily deleted attribute.
date:	created date of as attribute.
dateM:	not used currently.
owner:	owner of an attribute.
memo:	memo of an attribute. Useful for definition of attributes.
class:	class of an attribute. It might be "character" or "numeric".

Table "caseAttr" contatins information about attributes of cases.

variable:	name of case attributes, coresponding to name in attributes table.
value:	variable value.
caseID:	corresponding case id of a variable value.
date:	created date of a case attribute record.
dateM:	not used currently.
owner:	creator of the case attribute record.

Table "caselinkage" contatins information about the relationship between case and files of case.

caseid:	case id.
fid:	file id.
selfirst:	beginning position of a text segment associated with a case.
selend:	ending position of a text segment associated with a case..
status:	1 for standard status and 0 for temporarily deleted record.
owner:	creator of the case linkage.
date:	date of a created case linkage.
memo:	not used currently.

Table "cases" contatins information about case list.

name:	name of a case.
memo:	case memo.

owner: creator of a case.
date: date of creation of a case.
dateM: not used currently.
id: case id.
status: 1 for standard status and 0 for temporarily deleted record.

Table "codecat" contatins information about upper-level of code list.

name: name of code category.
cid: not used currently.
catid: id of code category.
owner: creator of code category.
date: date of creation of code category.
dateM: not used currently.
memo: code category memo.
status: 1 for standard status and 0 for temporarily deleted record.

Table "coding" contains information on codings.

cid : code id.
fid : file id.
seltext : a coding, that is the coded text segment.
selfirst : beginning position of the coded text segment.
selend : ending position of the coded text segment.
status : 1 for standard status. 0 for deleted codeings (for example when a code is deleted, the status of all associated coding
owner : name of coder or creator of a coding.
date : date of creation of a coding.
memo : coding memo.

Table "fileAttr" contatins information about attributes of files.

variable: charater, name of file attribute, coresponding to name in attributes table
value: value of the file attribute.
fileID: corresponding file id of the attribute.
date: created date of the file attribute.
dateM: not used currently.
owner: creator of the file attribute.

Table "filecat" contains information on the file categorization.

name: name of the file category.
fid: Not used.
catid: if of file category.

owner:	creator of file-category.
date:	date of creation of a file category.
dateM:	not used currently.
memo:	file category memo.
status:	1 for standard status and 0 for temporarily deleted record.

Table "freecode" contains information on the codes list.

name :	code name.
memo :	code memo.
owner :	creator of a code.
date :	date of creation of a code.
dateM :	not used currently.
id :	code id.
status :	1 for standard status and 0 for temporarily deleted record.
color:	color for code marker (added in version 0.19)

Table "image" contatins information about images. It is not used currently.

Table "imageCoding" contatins images coding. It is not used currently.

Table "journal" contatins information about field work journal. Journal titles are held in widget of ".JournalNamesWidget".

name:	name of a journal.
journal:	content of a journal.
date:	created date of a journal.
dateM:	not used currently.
owner:	owner of a journal.
status:	1 for standard status and 0 for temporarily deleted journal.

Table "project" contatins information about the project and *.rqda file.

encoding:	not used currently.
databaseversion:	version of RQDAtables.
date:	created date of the project.
dateM:	not used currently.
memo:	project memo.
BOM:	not used curently.
imageDir:	directory of image. Not used currently.
about:	meta information about the rqda file.

Table "source" contains the content of files. Files are held in widget of ".fnames_rqda".

name:	name of the file.

id:	id of the file.
file:	content of a file.
memo:	memo of the file.
owner:	creator the the file.
date:	the date of the file-import.
dataM:	date of last editing of the file content.
status:	1 for standard status and 0 for temporarily deleted file.

The "treecode" table contains information on the codes categorization (relationship between codes and the codecat). They are held in widget of ".CodeCatWidget". Codes of specific category are held in widget of ".CodeofCat".

cid:	code id.
catid:	code category id.
date:	date of creation of a code categorization.
dateM:	not used currently.
memo:	not used currently.
status:	1 for standard status and 0 for temporarily deleted file.
owner:	creator the the treecode.

Table "treefile" contatins information about file categorization (relation between source files and filecat).

fid:	file id.
catid:	file category id.
date:	date of creation of the file categorization.
dateM:	not used currently.
memo:	not used currently.
status:	1 for standard status and 0 for temporarily deleted record.
owner:	creator the the tree file.

Author(s)

HUANG Ronggui

`searchFiles` *Search files*

Description

Search files according to the pattern.

Usage

```
searchFiles(pattern, content = FALSE, Fid = NULL, Widget = NULL,is.UTF8 = FALSE)
```

Arguments

pattern	The criterion of search, see examples section for examples.
content	When it is TRUE, the content of files fitting the pattern will be returned as well.
Fid	integer vector, the ids of subset of files to search.
Widget	Character, name of a gtable widget. If it is not NULL, the file names fitting the pattern will pushed to that gtable widget using svalue method. One useful value is ".fnames_rqda", so the file names will be pushed to the Files Tab of RQDA. Others are ".FileofCat" and ".FileofCase".
is.UTF8	If the coding of pattern is UTF-8. If you are not sure, always use FLASE.

Details

This function use select statment of sql to search files (from source database table). The pattern is the WHERE clause (without the keyword WHERE). For more information, please refer to the website of SQLite syntax. All data in *.rqda use UTF-8 encoding, so the encoding of pattern matters. It will be converted to UTF-8 if it is not (is.UTF8=FALSE).

Value

A data frame with variables (which is invisible and you need to print it explicitly):

id	The file id.
name	The file name.
file	The file content. Only return when content is TRUE.

Author(s)

HUANG Ronggui

References

http://www.sqlite.org/lang_expr.html

See Also

gtable, localeToCharset

Examples

```
## Not run:
searchFiles("file like '%keyword%'")
## search for files who contain the word of "keyword"
searchFiles("file like 'keyword%'")
## search for files whose conent begin with the word of "keyword"
```

```
searchFiles("name like '%keyword'")
## search for files whose name end with the word of "keyword"
searchFiles("name like '%keyword one' and file like '%keyword tow%'")
## combined conidtions

## End(Not run)
```

---

summaryCodings *Summary of codings*

Description

Give a summary of codings of current project.

Usage

```
summaryCodings(byFile = FALSE, ...)
## S3 method for class 'summaryCodings'
print(x, ...)
```

Arguments

byFile	When it is FALSE, return the summary of current project. When it is TRUE, return the summary of coding for each coded file.
x	An object returned by summaryCoding.
...	Other possible arguments.

Value

A list:

NumOfCoding	Number of coding for each code.
AvgLength	Average number of characters in codings for each code.
NumOfFile	Number of files coded for each code.
CodingOfFile	Number of codings for each file. Returns NULL if byFile is FALSE.

Author(s)

HUANG Ronggui

See Also

getFileIds and getCodingTable

Examples

```
## Not run:
summaryCodings()
summaryCodings(FALSE)

## End(Not run)
```

`viewPlainFile` *View the file content of the selected file in File Widget without displaying the codings and annotations etc.*

Description

This function displays a data file in its bare form. The codings, annotations or other modifications done in RQDA won't be displayed.

Usage

```
viewPlainFile(FileNameWidget = .rqda$.fnames_rqda)
```

Arguments

`FileNameWidget` Users should leave it as it is.

Details

This function is useful to view the raw version of the data files. None of the codings, annotations, memos done in RQDA will be displayed on the file when it's called by this function. To use the function open a project and select the data file you wish to view and type 'ViewPlainFile()' in the command line.

Value

No value is return. It is used for the side effect: the function returns a widget window with the plain file in it.

Author(s)

HUANG Ronggui

`write.FileList` *Import a batch of files to the source table*

Description

If import individual file to the project, you can do it by clicking import button in the Files Tab. Sometimes, you want to import a batch of files quickly, you can do it by command. This function is used to import a batch of files into the source table in the *.rqda file.

Usage

```
write.FileList(FileList, encoding = .rqda$encoding, con = .rqda$qdacon, ...)
```

Arguments

`FileList`	A list. Each element of the list is the file content, and the `names(FileList)` are the respective file name.
`encoding`	Don't change this argument.
`con`	Don't change this argument.
`...`	`...` is not used.

Details

The file content will converted to UTF-8 character before write to *.rqda. The original content can be in any suitable encoding, so you can inspect the content correctly; In other words,the better practices is to used the corresponding encoding (you can get a hint by `localeToCharset` function) to save the imported files.

Value

This function is used for the side-effects. No value is return.

Author(s)

Huang Ronggui

Examples

```
## Not run:
Files <- list("File name one"="content of first File.",
              "File name two"="content of the second File.")
write.FileList(Files) ## Please launch RQDA(), and open a project first.

## End(Not run)
```

Index

∗Topic **package**
RQDA-package, 2
%and% *(*Ops.RQDA*)*, 22
%not% *(*Ops.RQDA*)*, 22
%or% *(*Ops.RQDA*)*, 22
%and%, *14*, *21*
%not%, *14*
%or%, *14*

and *(*Ops.codings*)*, 20

casesCodedBy, 3
casesCodedByAnd *(*casesCodedBy*)*, 3
casesCodedByNot *(*casesCodedBy*)*, 3
casesCodedByOr *(*casesCodedBy*)*, 3
CleanProject *(*Deletion*)*, 7
closeProject *(*Project*)*, 23
codingBySearch, 4
crossCodes, 5
crossTwoCodes *(*crossCodes*)*, 5

Deletion, 7

exportCodedFile, 8
exportCodings, 9

filesByCodes, 10
filesCodedBy, 10
filesCodedByAnd, *24*
filesCodedByAnd *(*filesCodedBy*)*, 10
filesCodedByNot, *24*
filesCodedByNot *(*filesCodedBy*)*, 10
filesCodedByOr, *24*
filesCodedByOr *(*filesCodedBy*)*, 10

getAttr, 11
getCaseIds *(*getCases*)*, 13
getCaseNames, *4*
getCaseNames *(*getCases*)*, 13
getCases, 13
getCodingsByOne, 14, *22*
getCodingTable, 15, *21*, *34*
getFileIds, *5*, *13*, *16*, *26*, *34*
getFileIds *(*getFiles*)*, 17
getFileIdSets, 16, *18*
getFileNames, *11*
getFileNames *(*getFiles*)*, 17
getFiles, *17*, 17
getMemos, 18
gregexpr, *4*, *5*
gselect.list, 19
gtable, *33*
gtext, *26*

intersect, *22*

list.deleted *(*Deletion*)*, 7
localeToCharset, *33*

nCodedByTwo, 20
not *(*Ops.codings*)*, 20

openProject *(*Project*)*, 23
Ops.codings, 20
Ops.RQDA, 22
or *(*Ops.codings*)*, 20

pdelete *(*Deletion*)*, 7
print.summaryCodings *(*summaryCodings*)*, 34
Project, 23

query, *27*
queryFiles, 24

relation, *6*, *21*, 25
retrieval, *16*, *18*, 26
RQDA *(*RQDA-package*)*, 2
RQDA-package, 2
RQDAQuery, 27
RQDATables, 28

searchFiles, 32
select.list, *19*
setdiff, *22*
showSubset *(*getAttr*)*, 11
summaryCodings, 34

undelete *(*Deletion*)*, 7
union, *22*

viewPlainFile, 35

write.FileList, 36