Package 'stacomirtools'

June 24, 2017

Version 0.5.2

Date 2017-06-10

Title 'ODBC' Connection Class for Package stacomiR

Author Cedric Briand [aut, cre]

Maintainer Cedric Briand <cedric.briand00@gmail.com>

Description S4 class wrappers for the 'ODBC' connection, also provides some utilities to paste small datasets to clipboard, rename columns. It is used by the package 'stacomiR' for connections to the database. Development versions of 'stacomiR' are available in R-forge.

License GPL (>= 2)

Collate 'ConnectionODBC.r' 'RequeteODBC.r' 'RequeteODBCwhere.r' 'RequeteODBCwheredate.r' 'utilities.r' 'stacomirtools.r'

LazyLoad yes

Depends RODBC

Imports methods,xtable,utils

Repository CRAN

Repository/R-Forge/Project stacomir

Repository/R-Forge/Revision 417

Repository/R-Forge/DateTimeStamp 2017-06-24 18:58:38

Date/Publication 2017-06-24 21:31:13 UTC

NeedsCompilation no

R topics documented:

`stacomirtools-package` *RODBC connector class and some utilities*

Description

This package contains S4 wrappers for 'ODBC' connection and some utilities

Details

Package:	stacomirtools
Type:	Package
Version:	0.5.2
Date:	2017-06-24
License:	GPL (>= 2)
LazyLoad:	yes

Author(s)

Cedric Briand <cedric.briand"at"eptb-vilaine.fr>

`chnames` *This function replaces the variable names in a data.frame*

Description

This function replaces the variable names in a data.frame

Usage

```
chnames(object, old_variable_name, new_variable_name)
```

Arguments

object	a data frame
old_variable_name	a character vector with old variables names
new_variable_name	a character vector with new variables names

Value

object

Author(s)

Cedric Briand <cedric.briand"at"eptb-vilaine.fr>

Examples

```
df <- data.frame("var1" = c("blue","red"), "var2" = c("nice","ugly"))
colnames(df) # "var1" "var2"
df <- chnames(object = df, old_variable_name = c("var1","var2"),
"new_variable_name" = c("color","beauty"))
colnames(df) # "color"  "beauty"
# the following will return an error, as the variable wrong_name is not in variable names
## Not run:
chnames(object = df, old_variable_name = c("wrong_name"),
"new_variable_name" = c("color")))

## End(Not run)
```

---

connect-methods *Methods for Function connect*

Description

see individual .r files for help and examples

Methods

`signature(object = "ConnectionODBC")` connect an 'ODBC' database,and eventually leaves it open for further queries, the connection may send message in the native language if 'stacomiR' package is in use

`signature(object = "RequeteODBC")` connect an 'ODBC' database,performs an sql request

`signature(object = "RequeteODBCwhere")` connect an 'ODBC' database,performs an sql request with where clause

`signature(object = "RequeteODBCwheredate")` connect an 'ODBC' database,performs an sql request with where clause for an interval

Examples

```
 showMethods("connect")
## Not run:
object<-new("RequeteODBCwhere")
connect(object)

## End(Not run)
```

ConnectionODBC-class *Class "ConnectionODBC"*

Description

Mother class for connection, opens the connection but does not shut it

Objects from the Class

Objects can be created by calls of the form `new("ConnectionODBC", ...)`.

`baseODBC`: Object of class `"vector"` The database

`silent`: Object of class `"logical"` The mode

`etat`: Object of class `"character"` The state

`connection`: Object of class `"ANY"` The connection

Slots

`baseODBC`: Object of class `"vector"` The database

`silent`: Object of class `"logical"` The mode

`etat`: Object of class `"character"` The state

`connection`: Object of class `"ANY"` The connection

Methods

connect `signature(object = "ConnectionODBC")`: Connection to the database

Note

Opens the connection but does not close it. This function is intended to be used with 'stacomiR' package, where the error message are collected from the database It has also been programmed to work without the 'stacomiR' package, as it will test for the existence of envir_stacomi environment.

Author(s)

cedric.briand"at"eptb-vilaine.fr

Examples

```
showClass("ConnectionODBC")
## Not run:
 ## this is the mother class, you don't have to use it,
 ## please use requeteODBC and daughter class instead
 object<-new("ConnectionODBC")
 object@baseODBC<-c("myODBCconnection","myusername","mypassword")
 object@silent<-FALSE
 object<-connect(object)
 odbcClose(object@connection)

## End(Not run)
```

---

ex *ex fonction to write to the clipboard*

Description

ex fonction to write to the clipboard

Usage

```
ex(d = NULL)
```

Arguments

d	a dataframe

Author(s)

Cedric Briand <cedric.briand"at"eptb-vilaine.fr>

---

funhtml *function used to print the html tables of output (see xtable documentation)*

Description

see **xtable** for further description, an xtable is created and printed to html format

Usage

```
funhtml(data,caption=NULL,top=TRUE,outfile=NULL,clipboard=FALSE,
append=TRUE,digits=NULL,...)
```

Arguments

data	a data frame
caption	the caption
top	a logical, if true the caption is placed on top
outfile	the path to the file
clipboard	if clipboard TRUE, a copy to the clipboard is made
append	is the file appended to the previous one ?
digits	the number of digits
...	additional parameters to be passed to the function

Value

an xtable

Author(s)

Cedric Briand <cedric.briand"at"eptb-vilaine.fr>

---

funout *Function loaded in this package to avoid errors, if the package is called without 'stacomiR'*

Description

This function will be replaced by a longer function using gWidgets if the package 'stacomiR' is loaded. It is provided there to avoid to pointing to an undefined global function. Normally the program tests for the existence of and environment envir_stacomi which indicates that the messages are to be displayed in the gWidget interface, so this code is to avoid notes in R.check.

Usage

```
funout(text,arret=FALSE,wash=FALSE)
```

Arguments

text	The text to display
arret	If true calls the program to stop and the message to be displayed
wash	Only used when called from within 'stacomiR', and there is a widget interface, kept there for consistency

Author(s)

Cedric Briand <cedric.briand"at"eptb-vilaine.fr>

induk *unique values of a vector*

Description

returns the index of values appearing only once in a vector : match(unique(a),a), replicated values are not returned on their second occurence

Usage

```
induk(a)
```

Arguments

a	a vector

Value

the index unique values within a vector

Author(s)

Cedric Briand <cedric.briand"at"eptb-vilaine.fr>

Examples

```
induk(c(1,1,2,2,2,3))
```

---

is.even *is.even function modified from package sma*

Description

is.even function modified from package sma (which did not verify that the entry was indeed an integer)

Usage

```
is.even(x)
```

Arguments

x	integer

Value

a logical

Author(s)

Adapted from Henrik Bengtsson

Examples

```
is.even(1)
is.even(2)
```

`is.odd` *id.odd function modified from package sma*

Description

id.odd function modified from package sma (which did not verify that the entry was indeed an integer)

Usage

```
is.odd(x)
```

Arguments

`x`	integer

Value

a logical

Author(s)

Adapted from Henrik Bengtsson

Examples

```
is.odd(1)
is.odd(2)
```

`killfactor` *very usefull function remove factor that appear, noticeably after loading with 'ODBC'*

Description

function used to remove factors that appear, noticeably after loading with 'ODBC'

Usage

```
killfactor(df)
```

Arguments

`df`	a data.frame

Value

df

Author(s)

Cedric Briand <cedric.briand"at"eptb-vilaine.fr>

Examples

```
df <- data.frame("var1" = c("blue","red"),"var2"=c("nice","ugly"))
df[,"var1"] <- as.factor(df[,"var1"])
df[,"var2"] <- as.factor(df[,"var2"])
df <- killfactor(df)
apply(df,1,function(x) is.factor(x)) # FALSE FALSE
```

`RequeteODBC-class` *Class "RequeteODBC"*

Description

'ODBC' Query. This class enables to retrieve data from the database. This class is inherited by RequeteODBCwhere and RequeteODBCwheredate

Objects from the Class

Objects can be created by calls of the form `new("RequeteODBC", sql=character(), query=data.frame())`.

`baseODBC`: Object of class `"vector"` The name, user and password of the database

`connection`: Object of class `"ANY"` The connection

`etat`: Object of class `"character"` The state of the query (Connecting, successful,...)

`silent`: Object of class `"logical"` True if the query must be executed silently, FALSE

`sql`: Object of class `"character"` The query

`query`: Object of class `"data.frame"` The result of the query

`open`: Object of class `"logical"` Should the connection remain open, choosing this ensures more rapid multiple queries

Extends

Class "ConnectionODBC", directly.

Methods

connect `signature(object = "RequeteODBC")`: Connection to the database

Note

Inherits from ConnectionODBC

Author(s)

cedric.briand"at"eptb-vilaine.fr

See Also

ConnectionODBC RequeteODBCwhere RequeteODBCwheredate

Examples

```
showClass("RequeteODBC")
## Not run:
 object=new("RequeteODBC")
 object@open=TRUE
 ## this will leave the connection open,
 ## by default it closes after the query is sent
 ## the following will work only if you have configured and 'ODBC' link
 object@baseODBC=c("myODBCconnection","myusername","mypassword")
 object@sql= "select * from mytable limit 100"
 object<-connect(object)
 odbcClose(object@connection)
 envir_stacomi=new.env()
 ## While testing I like to see the output of sometimes complex queries generated by the program
 assign("showmerequest",1,envir_stacomi)
 ## You can assign any values (here 1)
```

```
 ## just tests the existence of "showmerequest" in envir_stacomi
 object=new("RequeteODBC")
 object@baseODBC=c("myODBCconnection","myusername","mypassword")
 object@sql= "select * from mytable limit 100"
 object<-connect(object)
## the connection is already closed, the query is printed

## End(Not run)
```

RequeteODBCwhere-class

Class "RequeteODBCwhere"

Description

SQL Query with WHERE and ORDER BY clauses.

Objects from the Class

Objects can be created by calls of the form `new("RequeteODBCwhere", where=character(),and=vector(),order_by=ch`

`select`: Object of class `"character"` The "SELECT" part of the query

`where`: Object of class `"character"` The "WHERE" part of the query

`and`: Object of class `"vector"` The "AND" part of the query

`order_by`: Object of class `"character"` The "ORDER BY" part of the query

`sql`: Object of class `"character"` The query built by aggregating "select","where","and", and "order_by" slots

`query`: Object of class `"data.frame"` The result of the query

`open`: Object of class `"logical"` Should the connection remain open, choosing this ensures more rapid multiple queries

`baseODBC`: Object of class `"vector"` The name, user and password of the database

`silent`: Object of class `"logical"` TRUE if the query must be executed silently, FALSE else

`etat`: Object of class `"character"` The state of the query (Connecting, successful,...)

`connection`: Object of class `"ANY"` The database connection

Extends

Class `"RequeteODBC"`, directly. Class `"ConnectionODBC"`, by class "RequeteODBC", distance 2.

Methods

connect `signature(object = "RequeteODBCwhere")`: Connect to the database

Note

Inherits from RequeteODBC the syntax is where="WHERE ..." and =vector("AND...","AND...") order_by="ORDER BY.." The query will syntax will be printed upon failure.

Author(s)

cedric.briand"at"eptb-vilaine.fr

See Also

ConnectionODBC RequeteODBC RequeteODBCwheredate

Examples

```
showClass("RequeteODBCwhere")
## Not run:
 test<-0
 object=new("RequeteODBCwhere")
 object@baseODBC=c("myodbcconnection","myusername","mypassword")
 object@select= "select * from mytable limit 100"
 # assuming mycol, mycol1 and mycol2 are numeric
 object@where=paste(" where mycol>",test,sep="")
 object@and=paste(" and mycol2>",test," and mycol3<",test,sep="")
 object@order_by=" order by mycol1"
 object<-connect(object)
 ## now object@sql contains the syntax of the query.
 ## By changing the test variable, one can see how the
 ## function might be usefull
 ##  object@query contains the resulting data.frame

## End(Not run)
```

RequeteODBCwheredate-class

Class "RequeteODBCwheredate"

Description

Query with WHERE condition and overlaping dates clause.

Objects from the Class

Objects can be created by calls of the form `new("RequeteODBCwheredate", datedebut="POSIXlt",datefin="POSIXlt",`

`datedebut`: Object of class `"POSIXlt"` ~ The starting date

`datefin`: Object of class `"POSIXlt"` ~ The ending date

`colonnedebut`: Object of class `"character"` ~ The name begin column

`colonnefin`: Object of class `"character"` ~ The name end column

Slots

`datedebut`: Object of class `"POSIXlt"` ~ The starting date

`datefin`: Object of class `"POSIXlt"` ~ The ending date

`colonnedebut`: Object of class `"character"` ~ The name of the begin column

`colonnefin`: Object of class `"character"` ~ The name of the end column

`where`: Object of class `"character"` ~ The WHERE clause

`and`: Object of class `"vector"` ~ The AND clause

`order_by`: Object of class `"character"` ~ The ORDER BY clause

`sql`: Object of class `"character"` ~ The SELECT clause

`query`: Object of class `"data.frame"` ~ The result of the query

`baseODBC`: Object of class `"vector"` ~ The database

`silent`: Object of class `"logical"` ~ The mode

`etat`: Object of class `"character"` ~ The state

`connection`: Object of class `"ANY"` ~ The connection

Extends

Class `"RequeteODBCwhere"`, directly. Class `"RequeteODBC"`, by class "RequeteODBCwhere", distance 2. Class `"ConnectionODBC"`, by class "RequeteODBCwhere", distance 3.

Methods

connect `signature(object = "RequeteODBCwheredate")`: Connexion to the database

Note

Inherits from RequeteODBCwhere and uses its connect method with a new SetAs. This function is only usefull in databases supporting the "overlaps" statement.

Author(s)

cedric.briand"at"eptb-vilaine.fr

See Also

`ConnectionODBC` `RequeteODBC` `RequeteODBCwhere`

Examples

```
showClass("RequeteODBCwheredate")
```

tab2df *Function to transform a ftable into dataframe but just keeping the counts, works with ftable of dim 2*

Description

Function to transform a ftable into dataframe but just keeping the counts works with ftable of dim 2

Usage

```
tab2df(tab)
```

Arguments

tab	a flat table

Author(s)

Cedric Briand <cedric.briand"at"eptb-vilaine.fr>

Examples

```
df <- data.frame("var1" = c("blue","red"), "var2" = c("nice","ugly"))
ftdf <- ftable(df)
tab2df(ftdf)
```

Index