Package 'excel.link'

July 2, 2014

Type Package

Title Convenient way to work with data in Microsoft Excel

Version 0.6

Date 2013-09-04

Author Gregory Demin <excel.link.feedback@gmail.com>

Maintainer Gregory Demin <excel.link.feedback@gmail.com>

Depends RDCOMClient

OS_type windows

Description Allow dynamic access to data in Microsoft Excel. (e. g. xl[a1]=xl[b2]*3 and so on). Also class provided which allows treat data on Excel sheet similar to data.frame (indexing, subsetting). Microsoft Excel is required for this package.

License GPL (>= 2)

LazyLoad yes

Collate 'excel.link.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2013-09-05 11:53:48

R topics documented:

`current.graphics` *Auxilary function for export graphics to Microsoft Excel*

Description

If argument 'type' provided this function will save graphics from windows plotting device to temporary file and return path to this file. In other case (provided filename) 'type' will be ignored and resulted value is path to file with class attribute 'current.graphics'. So it could be used with expressions such xl[a1] = current.graphics(filename="plot.png").

Usage

```
current.graphics(type="emf",filename=NULL,...)
```

Arguments

`type`	file type. Ignored if argument 'filename' provided.
`filename`	filename (or full path) of file with graphics.
`...`	arguments for internally used savePlot function

Value

Path to file with saved graphics with class attribute 'current.graphics'. If used with argument 'type' than result has attribute 'temp.file'=TRUE

Author(s)

Gregory Demin <excel.link.feedback@gmail.com>

Examples

```
## Not run:
xl.workbook.add()
plot(sin)
xl[a1] = current.graphics()
plot(cos)
cos.plot = current.graphics()
xl.sheet.add()
xl[a1] = list("Cosine plotting",cos.plot,"End of cosine plotting")

# the same thing without graphic windows
```

```
png("1.png")
plot(sin)
dev.off()
sin.plot = current.graphics(filename = "1.png")
png("2.png")
plot(cos)
dev.off()
cos.plot = current.graphics(filename = "2.png")
output = list("Cosine plotting",cos.plot,"Sine plotting",sin.plot)
xl.workbook.add()
xl[a1] = output


## End(Not run)
```

`dimnames.excel.range` *Operations with column and row names of 'excel.range' and 'xl' classes.*

Description

Operations with column and row names of 'excel.range' and 'xl' classes

Usage

```
## S3 method for class 'excel.range'
dimnames(x)
## S3 method for class 'excel.range'
dim(x)

has.colnames(x)
has.rownames(x)
## Default S3 method:
has.colnames(x)
## Default S3 method:
has.rownames(x)
## S3 method for class 'excel.range'
has.colnames(x)
## S3 method for class 'excel.range'
has.rownames(x)
has.colnames(x) <- value
has.rownames(x) <- value
```

Arguments

x	xl or excel.range object
value	logical value

Details

'dimnames' read column/row names if xl/excel.range object has it. In opposite case it returns column/row names as in Excel (e. g. "a","b", "c").

'has.colnames' and 'has.rownames' sets/gets apropriate attributes of excel.range and xl objects. They are intended for internal usage.

Value

'dimnames' returns row and column names.

'dim' returns object row and column numbers

'has.rownames'/'has.colnames' - logical value. TRUE if excel.range/xl object has rownames/colnames.

Author(s)

Gregory Demin <excel.link.feedback@gmail.com>

Examples

```
## Not run:
xl.workbook.add()
xl[d4]=iris
test1=xl.connect.table("d4",row.names=FALSE,col.names=FALSE)
has.colnames(test1) # FALSE
has.rownames(test1) # FALSE
dimnames(test1)
rownames(test1)
colnames(test1)

xl.sheet.add()
xlrc[d4]=iris
test2=xl.connect.table("d4",row.names=TRUE,col.names=TRUE)
has.colnames(test2) # TRUE
has.rownames(test2) # TRUE
dimnames(test2)
rownames(test2)
colnames(test2)


## End(Not run)
```

---

xl *Object for working with Microsoft Excel data*

Description

Object aimed to work with data in Microsoft Excel ranges.

Usage

```
xl.selection(drop=TRUE,na="",row.names=FALSE,col.names=FALSE)

xl.current.region(str.rng,drop=TRUE,na="",row.names=FALSE,col.names=FALSE)

## S3 method for class 'xl'
x[str.rng,drop=!(has.rownames(x) | has.colnames(x)),na=""]
## S3 replacement method for class 'xl'
x[str.rng,na=""] <- value

## S3 method for class 'xl'
x[[str.rng,drop=!(has.rownames(x) | has.colnames(x)),na=""]]
## S3 replacement method for class 'xl'
x[[str.rng,na=""]] <- value

## S3 method for class 'xl'
x$str.rng
## S3 replacement method for class 'xl'
x$str.rng <- value
```

Arguments

xl	object representing Microsoft Excel application
x	xl, xlc, xlr or xlrc. xl - read/write without column and row names, "r" - with rownames, "c" - with colnames
str.rng	string which represents Excel range. For single bracket operations it can be without quotes in almost all cases
drop	logical. If TRUE the result is coerced to the lowest possible dimension. The default is to drop if there are no columns and rows names
row.names	a logical value indicating whether the Excel range contains the row names as its first column
col.names	a logical value indicating whether the Excel range contains the column names as its first row
na	character. NA representation in Excel. By default it is empty string
value	a suitable replacement value. It recycled to fill excel range only if value is object of length 1. In other cases size of excel range is ignored

Details

For convenient interactive usage arguments can be given without quotes in most cases (e. g. xl[a1]=5 or xl[u2:u85]="Hi" or xl[MyNamedRange]=42, but xl["Sheet1!A1"]=42). When it used in functions or you need to use variable as argument it is recommended apply double brackets notation:(e. g. xl[["a1"]]=5 or xl[["u2:u85"]]="Hi" or xl[["MyNamedRange"]]=42).

Difference between 'xl', 'xlc', 'xlrc' and 'xlr' is 'xl' ignore row and column names, 'xlc' suppose read and write to Excel with column names, 'xlrc' - with column and row names and so on. There is argument 'drop' which is TRUE by default for 'xl' and FALSE by default for other options.

'xl.selection' returns data.frame with data from current selection in Excel.

'xl.current.region' returns data.frame with data from current region (range which can be selected by pressing Ctrl+Shift+*) in Excel.

Value

Return appropriate dataset from Excel. Excel datetime type currently not supported.

Author(s)

Gregory Demin <excel.link.feedback@gmail.com>

Examples

```
## Not run:
data(iris)
rownames(iris) <- as.character(rownames(iris))
iris$Species <- as.character(iris$Species)
xl.workbook.add()
xlrc$a1 <- iris
xl.iris <- xl.current.region("a1",row.names=TRUE,col.names=TRUE)
identical(xl.iris,iris)

xl.sheet.add("Datasets examples")
data.sets <- list("Iris dataset",iris,"Cars dataset",cars,"Titanic dataset",as.data.frame(Titanic))
xlrc[a1] <- data.sets

## End(Not run)
```

---

xl.connect.table *Interactive work with data in Microsoft Excel*

Description

'xl.connect.table' returns object of 'excel.range' class which represent data on Excel sheet. This object can be treated similar to data.frame. So you can assign values, delete columns/rows and so on. For more information see details and examples.

'sort' sorts Excel range by single column (multiple columns currently not supported).

Usage

```
xl.connect.table(str.rng="A1",row.names=TRUE,col.names=TRUE,na="")

## S3 method for class 'excel.range'
x[i, j, drop = if (missing(i)) TRUE else !missing(j) && (length(j) == 1)]
## S3 method for class 'excel.range'
```

```
x$value

## S3 replacement method for class 'excel.range'
x[i,j] <- value
## S3 replacement method for class 'excel.range'
x$j <- value

## S3 method for class 'excel.range'
sort(x,decreasing=FALSE,column,...)
```

Arguments

x	An object of 'excel.range' class
str.rng	string which represents Excel range
i	rows to extract or replace. These are 'numeric' or 'character' or 'logical'
j	columns to extract or replace. These are 'numeric' or 'character' or 'logical'
drop	logical. If TRUE the result is coerced to the lowest possible dimension
row.names	a logical value indicating whether the Excel range contains the rows names as its first column
col.names	a logical value indicating whether the Excel range contains the columns names as its first row
na	character. NA representaion in Excel. By default it is empty string
value	a suitable replacement value
decreasing	logical. Should the sort be increasing or decreasing?
column	number or string. Column by which sort
...	arguments to be passed to or from methods or (for the default methods and objects without a class)

Details

Connected range is 'current region', e. g. selection wich can be obtained by pressing Ctrl+Shift+* when selected 'str.rng' (or top-left cell of this range is active).

Indices are 'numeric' or 'character' vectors or empty (missing). Numeric values are coerced to integer as by 'as.integer' (and hence truncated towards zero). Character vectors will be matched to the 'colnames' of the object (or Excel column names if has.colnames==FALSE). For extraction form if column name doesn't exist error will be generated. For replacement form new column will be created.

'i', 'j' can be logical vectors, indicating elements/slices to select. Such vectors are recycled if necessary to match the corresponding extent. 'i', 'j' can also be negative integers, indicating elements/slices to leave out of the selection.

Matrix indexing is not supported.

There is special value for 'column' argument in 'sort' - 'rownames'. In this case 'x' will be sorted by row names if it has it.

Value

For 'xl.connect.table' object of 'excel.range' class.

For '[' a data frame or a single column (the latter only when dimensions have been dropped). If non-existent column selected error will be generated.

For '$', a column.

For '[<-', '[[<-' and '$<-', object of 'excel.range' class.

'sort' invisibly return NULL.

Author(s)

Gregory Demin <excel.link.feedback@gmail.com>

Examples

```
## Not run:
### session example ###
library(excel.link)
xl.workbook.add()
xl.sheet.add("Iris dataset",before=1)
xlrc[a1] <- iris
xl.iris <- xl.connect.table("a1",row.names=TRUE,col.names=TRUE)
dists <- dist(xl.iris[,1:4])
clusters <- hclust(dists,method="ward")
xl.iris$clusters <- cutree(clusters,3)
plot(clusters)
pl.clus <- current.graphics()
cross <- table(xl.iris$Species,xl.iris$clusters)
plot(cross)
pl.cross <- current.graphics()
xl.sheet.add("Results",before=2)
xlrc$a1 <- list("Crosstabulation",cross,pl.cross,"Dendrogram",pl.clus)

### completly senseless actions #####
data(iris)
rownames(iris) <- as.character(rownames(iris))
iris$Species <- as.character(iris$Species)
xl.workbook.add()


xlrc[a1] <- iris
xl.iris <- xl.connect.table("a1",row.names=TRUE,col.names=TRUE)
identical(xl.iris[],iris)

iris <- iris[order(iris$Sepal.Length),]
sort(xl.iris,column="Sepal.Length")
identical(xl.iris[],iris)

sort(xl.iris,column="rownames")
iris <- iris[order(rownames(iris)),]
```

```
identical(xl.iris[],iris)

identical(xl.iris[,1:3],iris[,1:3])
identical(xl.iris[,3],iris[,3])
identical(xl.iris[26,1:3],iris[26,1:3])
identical(xl.iris[-26,1:3],iris[-26,1:3])
identical(xl.iris[50,],iris[50,])
identical(xl.iris$Species,iris$Species)
identical(xl.iris[,'Species',drop=FALSE],iris[,'Species',drop=FALSE])
identical(xl.iris[c(TRUE,FALSE),'Sepal.Length'],iris[c(TRUE,FALSE),'Sepal.Length'])

xl.iris[,'group'] <- xl.iris$Sepal.Length>mean(xl.iris$Sepal.Length)
iris[,'group'] <- iris$Sepal.Length>mean(iris$Sepal.Length)
identical(xl.iris[],iris)

xl.iris$temp <- c('aa','bb')
iris$temp <- c('aa','bb')
identical(xl.iris[],iris)

xl.iris[,"temp"] <- NULL
iris[,"temp"] <- NULL
identical(xl.iris[],iris)



## End(Not run)
```

`xl.get.excel` *Function which return reference to Excel application*

Description

Returns reference to Microsoft Excel application. If there is no instance already in existence it will create a new instance.

Usage

```
xl.get.excel()
```

Value

An object of class 'COMIDispatch' (as returned by 'COMCreate' from RDCOMClient package).

Author(s)

Gregory Demin <excel.link.feedback@gmail.com>

Examples

```
## Not run:
xls=xl.get.excel()


## End(Not run)
```

`xl.read.file, xl.save.file`

Functions for saving and reading data to/from Excel file.

Description

Functions for saving and reading data to/from Excel file.

Usage

```
xl.read.file(filename, header = TRUE, row.names = NULL, col.names = NULL,
           xl.sheet = NULL, top.left.cell = "A1", na = "", excel.visible = FALSE)

xl.save.file(r.obj, filename, row.names = TRUE, col.names = TRUE,
           xl.sheet = NULL, top.left.cell = "A1", na = "", excel.visible = FALSE)
```

Arguments

`filename`	a character string naming a file
`r.obj`	R object
`header`	a logical value indicating whether the file contains the names of the variables as its first line. If TRUE and top-left corner is empty cell, first column is considered as row names. Ignored if row.names or col.names is not NULL.
`row.names`	a logical value indicating whether the row names of r.obj are to be read/saved along with r.obj
`col.names`	a logical value indicating whether the column names of r.obj are to be read/saved along with r.obj
`xl.sheet`	character. Name of Excel sheet where data is located/will be saved. By default it is NULL and data will be read/saved from/to active sheet.
`top.left.cell`	character. Top-left corner of data in Excel sheet. By default is 'A1'.
`na`	character. NA representation in Excel. By default it is empty string
`excel.visible`	a logical value indicating will Excel visible during this operations. FALSE by default.

Details

xl.read.file reads only rectangular data set. It is highly recommended to have all column names and ids in data set. Orphaned rows/columns located apart from the main data will be ignored. xl.save.file can save all objects for which xl.write method exists - see example.

Value

xl.read.file always return data.frame. xl.save.file return NULL.

Author(s)

Gregory Demin <excel.link.feedback@gmail.com>

See Also

xl.write, xl.workbook.save, xl.workbook.open, current.graphics

Examples

```
## Not run:
data(iris)
xl.save.file(iris,"iris.xlsx")
xl.iris = xl.read.file("iris.xlsx")
all(iris == xl.iris) # Shoud be TRUE
unlink("iris.xlsx")

# Save to file list with different data types
dists = dist(iris[,1:4])
clusters = hclust(dists,method="ward")
iris$clusters = cutree(clusters,3)
png("1.png")
plot(clusters)
dev.off()
pl.clus = current.graphics(filename="1.png")
cross = table(iris$Species,iris$clusters)
png("2.png")
plot(cross)
dev.off()
pl.cross = current.graphics(filename="2.png")
output = list("Iris",pl.clus,cross,pl.cross,"Data:","",iris)
xl.save.file(output,"output.xls")
xl.workbook.open("output.xls")
# xl.workbook.close() # close workbook
# unlink("output.xls") # delete file


## End(Not run)
```

`xl.sheets` *Basic operations with worksheets*

Description

Functions for basic operations with worksheets

Usage

```
xl.sheets()
xl.sheet.add(xl.sheet.name=NULL,before=NULL)
xl.sheet.delete(xl.sheet=NULL)
xl.sheet.activate(xl.sheet)
```

Arguments

`xl.sheet.name`	sheet name in active workbook
`before`	sheet name or sheet number in active workbook before which new sheet will be added
`xl.sheet`	sheet name or sheet number in active workbook

Details

'xl.sheet.delete' deletes sheet with given name/number. If name doesn't submitted it delete active sheet.

'xl.sheet.add' adds new sheet with given name and invisibly returns name of this newly added sheet. Added sheet become active. If 'xl.sheet.name' is missing default name will be used. If 'before' argument is missing, sheet will be added at the last position. If sheet with given name already exists error will be generated.

'xl.sheet.activate' activates sheet with given name/number. If sheet with this name doesn't exist error will be generated.

Value

'xl.sheets' returns vector of sheet names in active workbook.

'xl.sheet.add'/'xl.sheet.activate' invisibly returns name of created/activated sheet.

Author(s)

Gregory Demin <excel.link.feedback@gmail.com>

See Also

xl.workbooks

Examples

```
## Not run:
xl.workbook.add()
sheets <- xl.sheets()
xl.sheet.add("Second")
xl.sheet.add("First",before="Second")
for (sheet in sheets) xl.sheet.delete(sheet) # only 'First' and 'Second' exist in workbook now
xl.sheet.activate("Second") #last sheet activated


## End(Not run)
```

---

xl.workbooks *Basic operations with Excel workbooks*

Description

Functions for basic operations with Excel workbooks

Usage

```
xl.workbooks()
xl.workbook.add(filename=NULL)
xl.workbook.open(filename)
xl.workbook.close(xl.workbook.name=NULL)
xl.workbook.save(filename)
xl.workbook.activate(xl.workbook.name)
```

Arguments

`filename`	Excel workbook filename
`xl.workbook.name`	Excel workbook name

Details

'xl.workbook.close' closes workbook with given name. If name doesn't submitted it closed active workbook. It doesn't prompt about saving so if you don't save changes before closing all changes will be lost.

'xl.workbook.save' saves active workbook. If only filename submitted it saves in the working directory. If name of workbook is omitted than new workbook is saved under its default name in the current working directory. It doesn't prompt about overwriting if file already exists.

'xl.workbook.add' adds new workbook and invisibly returns name of this newly created workbook. Added workbook become active. If 'filename' argument provided then Excel workbook 'filename' will be used as template.

'xl.workbook.activate' activates workbook with given name. If workbook with this name doesn't exists error will be generated.

'xl.workbooks' returns character vector of names of all opened workbooks.

Value

'xl.workbooks' returns character vector of open workbooks. 'xl.workbook.save' invisibly returns path to the saved workbook 'xl.workbook.add'/'xl.workbook.activate'/'xl.workbook.activate' invisibly returns name of created/activated workbook.

Author(s)

Gregory Demin <excel.link.feedback@gmail.com>

See Also

xl.sheets

Examples

```
## Not run:

## senseless actions
xl.workbook.add()
xlrc[a1] <- iris
xl.workbook.save("iris.xlsx")
xl.workbook.add()
xlrc[a1] <- cars
xl.workbook.save("cars.xlsx")
xl.workbook.activate("iris")
xl.workbook.close("cars")
xl.workbook.open("cars.xlsx")
xl.workbooks()
for (wb in xl.workbooks()) xl.workbook.close(wb)
unlink("iris.xlsx")
unlink("cars.xlsx")


## End(Not run)
```

---

xl.write *Methods for writing data to Excel sheet*

Description

Methods for writing data to Excel sheet

Usage

```
xl.write(r.obj,xl.rng,na="",...)
## Default S3 method:
xl.write(r.obj,xl.rng,na="",row.names=TRUE,...)
## S3 method for class 'data.frame'
xl.write(r.obj,xl.rng,na="",row.names=TRUE,col.names=TRUE,...)
## S3 method for class 'matrix'
xl.write(r.obj,xl.rng,na="",row.names=TRUE,col.names=TRUE,...)
## S3 method for class 'factor'
xl.write(r.obj,xl.rng,na="",row.names=TRUE,...)
## S3 method for class 'list'
xl.write(r.obj,xl.rng,na="",...)
## S3 method for class 'table'
xl.write(r.obj,xl.rng,na="",...)
## S3 method for class 'current.graphics'
xl.write(r.obj,xl.rng,na="",delete.file=FALSE,...)
```

Arguments

r.obj	R object
xl.rng	An object of class 'COMIDispatch' (as used in RDCOMClient package) - reference to Excel range
na	character. NA representaion in Excel. By default it is empty string
row.names	a logical value indicating whether the row names of r.obj are to be written along with r.obj
col.names	a logical value indicating whether the column names of r.obj are to be written along with r.obj
delete.file	a logical value indicating whether delete file with graphic after insertion in Excel
...	arguments for further processing

Details

'xl.rng' should be COM-reference to Excel range, not string. Method invisibly returns number of columns and rows occupied by 'r.obj' on Excel sheet. It's useful for multiple objects writing to prevent their overlapping.

It is more convenient to use 'xl' object. 'xl.write' aimed mostly for programming purposes, not for interactive usage.

Value

c(rows,columns)

Invisibly returns rows and columns number ocuppied by r.obj on Excel sheet.

Author(s)

Gregory Demin <excel.link.feedback@gmail.com>

See Also

xl, xlr,xlc,xlrc, current.graphics

Examples

```
## Not run:
xls=xl.get.excel()
xl.workbook.add()
rng=xls[["Activesheet"]]$Cells(1,1)
nxt=xl.write(iris,rng,row.names=TRUE,col.names=TRUE)
rng=rng$Offset(nxt[1]+1,0)
nxt=xl.write(cars,rng,row.names=TRUE,col.names=TRUE)
rng=rng$Offset(nxt[1]+1,0)
nxt=xl.write(as.data.frame(Titanic),rng,row.names=TRUE,col.names=TRUE)

xl.sheet.add()
rng=xls[["Activesheet"]]$Cells(1,1)
data.sets=list("Iris dataset",iris,"Cars dataset",cars,"Titanic dataset",as.data.frame(Titanic))
xl.write(data.sets,rng,row.names=TRUE,col.names=TRUE)


## End(Not run)
```

Index