Package 'reach'

June 21, 2016

Title Improving interoperability between R and MATLAB

Version 0.4.5

Author Christoph Schmidt <christoph.schmidt@med.uni-jena.de>

Maintainer Christoph Schmidt <christoph.schmidt@med.uni-jena.de>

Description An implementation of utility functions for improving R < > MATLAB interoperability.

SystemRequirements Matlab

Depends R (>= 3.2.1)

Imports stringr (>= 1.0.0), R.matlab (>= 3.2.0)

Suggests testthat (>= 0.10.0)

LazyData true

License MIT + file LICENSE

URL https://github.com/schmidtchristoph/reach

BugReports https://github.com/schmidtchristoph/reach/issues

RoxygenNote 5.0.1

R topics documented:

convert2RData *Converts a Matlab data file (.mat) to R data file (.RData)*

Description

Converts all or specified Matlab .mat files in the specified directory to .RData files, while keeping the .mat files unchanged. The Matlab files must have been saved in a MAT file format version supported by R.matlab's `readMat` function (e.g. saved with the -v7 option flag, but not with the -v7.3 flag).

Usage

```
convert2RData(matfile = NULL, dir = "./", verbose = FALSE)
```

Arguments

`matfile`	character or character vector denoting one or several matfiles, e.g. "mymatlabdata", "mymatlabdata.mat", c("one.mat", "two.mat"). Providing the file type specifier ".mat" is optional. Defaults to NULL, which means that all .mat files in the specified directory `dir` will be converted.
`dir`	path to a directory which contains one or several .mat files that should be converted to .RData files. Defaults to the current working directory.
`verbose`	logical indicating whether console output about the conversion process should be printed

Author(s)

Christoph Schmidt <christoph.schmidt@med.uni-jena.de>

Examples

```
## Not run:

##### conversion: specified .mat file in current working directory ####
v <- sample(1:10,4)
m <- matrix(runif(9),3,3)
R.matlab::writeMat("file_convert2RData.mat", v=v, m=m)
rm(v,m)
convert2RData("file_convert2RData.mat")
load("file_convert2RData.RData")
print(v)
print(m)
file.remove(c("file_convert2RData.RData", "file_convert2RData.mat"))



#### conversion: all .mat files in a specified directory ####
this_dir <- getwd()
m   <- matrix(runif(9),3,3)
v   <- seq(1,100)
R.matlab::writeMat("dir_convert2RData_1.mat", m=m)
R.matlab::writeMat("dir_convert2RData_2.mat", v=v)
```

```
rm(v,m)
convert2RData(dir = this_dir, verbose = TRUE)
load(paste(this_dir, "/dir_convert2RData_1.Rdata", sep = ""))
print(m)
load(paste(this_dir, "/dir_convert2RData_2.Rdata", sep = ""))
print(v)
file.remove(c(paste(this_dir, "/dir_convert2RData_1.mat", sep = ""),
   paste(this_dir, "/dir_convert2RData_2.mat", sep = ""),
   paste(this_dir, "/dir_convert2RData_1.Rdata", sep = ""),
   paste(this_dir, "/dir_convert2RData_2.Rdata", sep = "")))



#### conversion: specified .mat file in a specified directory ####
this_dir <- getwd()
v   <- seq(1,10)
R.matlab::writeMat("file_dir_convert2RData.mat", v=v)
rm(v)
convert2RData("file_dir_convert2RData.mat", this_dir)
load("file_dir_convert2RData.RData")
print(v)
file.remove(c("file_dir_convert2RData.mat", "file_dir_convert2RData.RData"))



#### conversion: several specified .mat files in current working directory ####
v <- sample(1:10,4)
m <- matrix(runif(9),3,3)
R.matlab::writeMat("twofiles_convert2RData_1.mat", v=v)
R.matlab::writeMat("twofiles_convert2RData_2.mat", m=m)
rm(v,m)
convert2RData(c("twofiles_convert2RData_1.mat", "twofiles_convert2RData_2.mat"))
load("twofiles_convert2RData_1.RData")
print(v)
load("twofiles_convert2RData_2.RData")
print(m)
file.remove(c("twofiles_convert2RData_1.mat", "twofiles_convert2RData_2.mat",
   "twofiles_convert2RData_1.RData", "twofiles_convert2RData_2.RData"))

## End(Not run)
```

`getMacMatlab` *Returns the name of the Matlab app of the latest version in the OSX Applications folder*

Description

Utility function that returns the latest Matlab app (with the highest version number) in the OSX Applications folder, e.g. "MATLAB_R2014a.app".

Usage

```
getMacMatlab()
```

Value

Latest Matlab application on Mac OSX

Author(s)

Christoph Schmidt <christoph.schmidt@med.uni-jena.de>

`getMatlabCall` *Get OS-specific Matlab program call string*

Description

Returns the matlab programm name for OS X, Linux and Windows.

Usage

```
getMatlabCall()
```

Author(s)

Christoph Schmidt <christoph.schmidt@med.uni-jena.de>

`isvector` *Checks whether input variable is a vector*

Description

An input is considered to be a vector if it has dimensions (n,1) or (1,n), n>1. Returns TRUE if the input is a vector according to this definition. Therefore, input that is a one-dimensional 'matrix' in R (is.matrix = TRUE and is.vector = FALSE) would also be regarded as a vector.

Usage

```
isvector(obj)
```

Arguments

`obj`	Data whose type is being tested to be a vector (n,1) or (1,n), n>1

Value

This function returns a boolean indicating whether input obj is a vector

Author(s)

Christoph Schmidt <christoph.schmidt@med.uni-jena.de>

Examples

```
v <- c(1, 2, 3)
reach:::isvector(v)

reach:::isvector(t(v))

m <- matrix(1:4, 2, 2)
reach:::isvector(m)

v <- matrix(1:4, 4, 1)
reach:::isvector(v)
```

`isWin` *Is the underlying OS Windows?*

Description

Returns a logical that indicates whether **reach** is used on a Windows machine (which is useful information for adjusting path specifications).

Usage

```
isWin()
```

Author(s)

Christoph Schmidt <christoph.schmidt@med.uni-jena.de>

`matlabExportList` *Reformats an R list to be exported to Matlab as cell-array*

Description

Exporting a R list with unnamed entries to Matlab using the R.matlab::writeMat function yields a Matlab struct with no fields (an empty struct). With the help of the matlabExportList function such a R list is reformated so that the export results in a struct with fields (1,2,...), accessible with the Matlab getfield() function. In Matlab, the loaded struct can then be further processed with the Matlab function 'rList2Cell()', which is distributed with this package, to yield a Matlab cell-array. For export, the package 'R.matlab' has to be used. Note that in particular for storing a multidimensional matrix one can also use R arrays instead of lists (a<-array(dim=c(3,3,2); a[,,1]<-matrix(99,3,3); ...) which will be exported just fine to Matlab and don't need any further processing.

Usage

```
matlabExportList(rlist)
```

Arguments

`rlist`	List that is exported to Matlab as cell-array

Details

A list containing lists is not supported by the writeMat() function and provokes an error. Consequently, this is checked for in matlabExportList and triggers an error.

Value

This function returns a reformated list that can be exported to Matlab with writeMat(). In Matlab it should be transformed to a cell-array using the Matlab function 'rList2Cell()', which is distributed with this package.

Author(s)

Christoph Schmidt <christoph.schmidt@med.uni-jena.de>

See Also

rList2Cell

Examples

```
## Not run:

rlist <- list(matrix(sample(100,16),4,4), c(1,2,3,4), "somestring")
print(rlist)
matlablist <- matlabExportList(rlist)
print(matlablist)
R.matlab::writeMat("test.mat", myexportdata=matlablist)
# in Matlab or using runMatlabCommand() (and having "rList2Cell.m" on the Matlab path):
runMatlabCommand("load test.mat; myexportdata, rc=rList2Cell(myexportdata); celldisp(rc); quit")
system("rm test.mat")

## End(Not run)
```

---

matWrite *Writes .mat files for exporting data to be used with Matlab*

Description

Writes .mat files to store R session data using the R.matlab package and takes care that logicals and atomic vectors are saved properly: currently, R.matlab does not write logicals and atomic vectors (not 1D arrays/ matrices) in a way that they can be opened properly in Matlab (logicals will not be stored and atomic vectors will be transposed in the Matlab session - but they appear untransposed when read back from the .mat file into R using R.matlab::readMat()). This function is a convenient wrapper for R.matlab::writeMat() that stores logicals as 0 / 1 and that transposes atomic vectors when saving the matfile.

Usage

```
matWrite(fn, vars)
```

Arguments

fn	file name, a character string
vars	character vector containing a comma separated listing of variables to be saved in the matfile. The variables have to exist in the environment where the function was called from.

Author(s)

Christoph Schmidt <christoph.schmidt@med.uni-jena.de>

Examples

```
A       <- matrix(c(2,3,4,2, 1,1,2,6, 8,3,9,7), 3, 4, byrow = TRUE)
b       <- c(3, 5, 1, 9, 18, 2) # atomic vector 1x6
cc      <- array(b, c(1, length(b))) # array vector 1x6
dd      <- t(cc) # array vector 6x1
myChar  <- c("from", "R", "to", "Matlab")
bool    <- TRUE # logical
k       <- FALSE
l       <- FALSE
fn      <- "mytestmat"
vars    <- "A, b, cc, dd, myChar, bool, k, l"
matWrite(fn, vars)
unlink(paste(fn, ".mat", sep = ""))
```

---

rList2Cell — *Conversion of a R list, which was processed with the R function "matlabExportList()" and then imported to Matlab to a Matlab cell-array*

Description

This is a Matlab function. It transforms a R list datatype (which is imported in Matlab as a struct) to a Matlab cell-array. Also recovers/ reformats multi-arrays contained in this R list (which are only exported as vectors).

Usage

```
rList2Cell(importlist)
```

Arguments

importlist	the imported struct equivalent of the R list, which was reformated in R using matlabExportList.R and exported to Matlab using writeMat() from the R.matlab package

Value

A Matlab cell-array containing in each cell the corresponding element of the original R list data (before it was reformated using matlabExportList.R()); also multi-arrays stored in the original R list datatype are recovered

Author(s)

Christoph Schmidt <christoph.schmidt@med.uni-jena.de>

See Also

matlabExportList

`runMatlabCommand` *Starts Matlab on the R console and executes one or several input Matlab commands*

Description

Starts Matlab on the R console and let it executes the input Matlab command or several input commands, like function calls (separated by ";") and quits Matlab. No values are directly returned back to the R session. To achieve this, use the function `runMatlabFct()`. Discerns the OS X, Linux and MS Windows Matlab app shell command. Automatically changes to the current R working directory in Matlab so that .mat files with Matlab results would be saved there instead of the default Matlab working directory.

Usage

```
runMatlabCommand(commandName, verbose = FALSE, do_quit = TRUE)
```

Arguments

`commandName`	a string denoting the Matlab command/ commands
`verbose`	logical indicating if the final internally generated Matlab command should be printed to the R console
`do_quit`	logical indicating if the Matlab process should quit itself. The default is TRUE, however, if the Matlab command is a plot function then one wants Matlab to keep displaying the plot window and not quit. This means the user has to quit Matlab manually prior to continue working in the current R session.

Details

As R and Matlab cannot directly exchange data natively, no value can be returned. Instead, let Matlab save the results of its computations and load these into R for further processing. An error in the Matlab command prevents Matlab from quitting in the R console and might require killing the Matlab process or an re-start of the R session. (You migth want to check the command in Matlab before executing it within R.) The commandName could look something like this: "load someData.mat; [out1, out2]=someMatlabFunction(in1, in2, in3); save someData2.mat; quit"

Author(s)

Christoph Schmidt <christoph.schmidt@med.uni-jena.de>

See Also

runMatlabFct, runMatlabScript

Examples

```
## Not run:

commandName <- "x=1:2:7; y=3; disp(x); disp(x.^y); quit"
runMatlabCommand(commandName)



commandName2 <- "M=magic(4); disp(M); eig(M)"
runMatlabCommand(commandName2)



wrong_but_corrected_commandName <- "M=magic(4); disp(M); eig(M) quit"
runMatlabCommand(wrong_but_corrected_commandName)



commandName3 <- "A=magic(3); save('magic.mat', 'A', '-v7'); quit"
runMatlabCommand(commandName3)
input        <- R.matlab::readMat("magic.mat")
print(input$A)
invisible(capture.output(file.remove("magic.mat")))



# !the Matlab process has to be terminated manually!
commandName4 <- "A=magic(40); imagesc(A)"
runMatlabCommand(commandName4, do_quit = FALSE)



# !the Matlab process has to be terminated manually!
commandName4 <- "spy; quit" # quit will be internally removed
runMatlabCommand(commandName4, do_quit = FALSE)


## End(Not run)
```

`runMatlabFct` *Runs a Matlab function like an R function and returns its results*

Description

Runs Matlab on the R console, evaluates the specified Matlab function with given arguments and directly returns the Matlab output as a single value or a list of values back to the R session. The function acts as a proxy for the specified Matlab function. It handles all Matlab related things internally and transparently to the user and just returns the specified Matlab output arguments as results of the computations performed by Matlab.

Usage

```
runMatlabFct(fcall)
```

Arguments

fcall	string specifying the Matlab function call with output and input parameters, just as one would call the function inside a regular Matlab session. The input arguments specified in fcall must be variable names stored in the current R session (the environment where the function was called from) or numeric values. Variable names are preferred and should be the standard for using this function. If no output parameter are contained in fcall, then it is assumed that the user expects Matlab to not automatically quit (e.g. because the function call was a plotting function and the plot window should stay open). In this case the Matlab process has to be terminated manually (!) by the user before the function can terminate and one can continue to work in the R session. Nested Matlab function calls (function as input argument for a function) are currently not supported.

Details

This function calls the user specified Matlab function and manages the necessary data exchange transparently, thus providing a seamless experience. The data is exchanged robustly over the file system, using temporary files that will be deleted automatically.

Value

The results of the Matlab function call. A list of named entries that correspond to the output arguments of the Matlab function as specified in *fcall*.

Note

For starting several Matlab functions independently of each other, or a chain of Matlab functions, consider using the 'runMatlabCommand()' function, or writing a Matlab script file and use the 'runMatlabScript()' function and store the results in a .mat file that can be read back into R with the help of the 'convert2RData()' function.

Author(s)

Christoph Schmidt <christoph.schmidt@med.uni-jena.de>

See Also

runMatlabScript, runMatlabCommand

Examples

```
## Not run:
a       <- c(1,2,1,4,1,5,4,3,2,2,1,6,3,1,3,5,5)
b       <- c(4,6,9)
fcall   <- "[bool, pos] = ismember(b,a)"

results <- runMatlabFct(fcall)

bool    <- results$bool
pos     <- results$pos
print(a)
print(b)
print(fcall)
print(bool)
```

```
print(pos)



# !the Matlab process has to be terminated manually!
runMatlabFct("image")



# !the Matlab process has to be terminated manually!
# wrong Matlab function input ( it should be penny or penny() ), but corrected internally
runMatlabFct("penny(")



M       <- matrix(c(2,-1,0,  -1,2,-1,  0,2,3), 3, 3)
fcall   <- "C = chol(M)"
results <- runMatlabFct(fcall)
print(results)



A  <- runMatlabFct("A=rand(6)")$A
A_ <- runMatlabFct("A_=inv(A)")$A_
print(round(A %*% A_))



nu_  <- 0.32
vec_ <- c(1,2,5,2,6)
res_ <- runMatlabFct("v_ = bessely(nu_, vec_)")
print(res_$v_)



orig_str <- 'this_test_was_my_first_test'
old_sub  <- 'test'
new_sub  <- 'assignment'
new_str  <- runMatlabFct('str=strrep(orig_str, old_sub, new_sub)')
print(new_str$str)


## End(Not run)
```

`runMatlabScript` *Starts Matlab on the R console and executes a Matlab script file*

Description

Starts Matlab on the R console (or for MS Windows: in an extra Matlab console window), executes the input Matlab script file (.m file) and quits Matlab.

Usage

```
runMatlabScript(scriptName)
```

Arguments

scriptName	String denoting the .m script (with or without the file extension)

Details

As R and Matlab cannot directly exchange data natively, no value will be returned directly. Instead, let Matlab save the results of its computations and load these into R for further processing. See also the following system call example: system('/Applications/MATLAB_R2013a.app/bin/matlab -nosplash -nodesktop -r "S_test; quit;"') An error in the Matlab script prevents Matlab from quitting in the R console and might require killing the Matlab process or a re-start of the R session. So check the script in Matlab before executing it within R.

Note

The function expects the script to be saved in the current R working directory. The script file might as well be generated by R code on the fly as shown in the examples section.

Author(s)

Christoph Schmidt <christoph.schmidt@med.uni-jena.de>

See Also

convert2RData, runMatlabCommand

Examples

```
## Not run:

scriptName <- "myscript.m"
mypath <- getwd()
print(mypath)
scriptCode <- "pwd, x=1:2:7; y=3; z=x.^y; save xyz.mat x y z -v7"
writeLines(scriptCode, con=scriptName)
list.files(mypath)

runMatlabScript(scriptName)

list.files(mypath)
system(paste("rm ", scriptName, sep=""))
inp <- R.matlab::readMat("xyz.mat")
str(inp)
system("rm xyz.mat")
list.files(mypath)

## End(Not run)
```

`str_extractCommaSepArgs` *Extract arguments given in a string, separated by commas*

Description

Parses an input string and returns the comma separated arguments.

Usage

```
str_extractCommaSepArgs(args_str)
```

Arguments

`args_str`	character vector containing (variable) names separated by commas

Value

Returns a list containing the (variable) names (characters) stored in `args_str`.

Author(s)

Christoph Schmidt <christoph.schmidt@med.uni-jena.de>

Examples

```
vars    <- "A, B, myChar, level, k,  test, te_st  , newval"
reach:::str_extractCommaSepArgs(vars)
```

Index