Package 'matconv'

May 13, 2021

Type Package

Title A Code Converter from the Matlab/Octave Language to R

Version 0.4.2

Maintainer Siddarta Jairam <sidjsb@gmail.com>

Description Transferring over a code base from Matlab to R is often a repetitive and inefficient use of time. This package provides a translator for Matlab / Octave code into R code. It does some syntax changes, but most of the heavy lifting is in the function changes since the languages are so similar. Options for different data structures and the functions that can be changed are given. The Matlab code should be mostly in adherence to the standard style guide but some effort has been made to accommodate different number of spaces and other small syntax issues. This will not make the code more R friendly and may not even run afterwards. However, the rudimentary syntax, base function and data structure conversion is done quickly so that the maintainer can focus on changes to the design structure.

License GPL (>= 2)

Imports methods

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation no

Author Siddarta Jairam [aut, cre],
David Hiebeler [ctb]

Repository CRAN

Date/Publication 2021-05-13 04:20:07 UTC

R topics documented:

`makeDataMap` *Make the maps for the data*

Description

Make the maps for the data

Usage

```
makeDataMap(leftSym, rightSym, rClass, matClass = "")
```

Arguments

`leftSym`	The left symbol that contains the Matlab data
`rightSym`	the right symbol that contains the Matlab data
`rClass`	The formal r class name that defines what the R data is outputted as
`matClass`	The name of the Matlab class that should be converted

Details

The requirements for conversion are the bounds given by both left and right symbols or the MatLab class. The Matlab class allows for the conversion of structures but is really just a dictionary for the different bounds.

Value

A function that takes in a Matlab lines and changes the data into R data lines

Examples

```
dataMap <- makeDataMap("[", "]", "matrix")
dataMap("thing <- [23,2, 3.2; 7, 6, 8]")
# "thing <- matrix(c(23, 2, 3.2, 7, 6, 8), nrow = 2, ncol = 3)"

dataMap <- makeDataMap(rClass = "list", matClass = "cell")
dataMap("otherThing <- {23,2, '3.2'; NaN, 6, 8}")
# "otherThing <- list(list(23, 2, '3.2'), list(NaN, 6, 8))"
```

makeFuncMaps *Turn dictionary lines into functions that map matlab to R function calls*

Description

Turn dictionary lines into functions that map matlab to R function calls

Usage

```
makeFuncMaps(addDict = NULL, pathDict = "")
```

Arguments

addDict	An optional character vector with manufactored lines
pathDict	The path to a text file with the dictionary lines written to it

Details

The output of the individual maps consits of the actual map for the given matlab arguments as a vector and a list of flags included in the dictionary. The argMap itself is a list of potential functions that could be used if a some flags are detected in the dictionary line. A more expansive look at the different dictionaries that could be used can be seen in the base dictionary at "extdata/HiebelerDict.txt" or in the vignette "vignettes/functionCalls.rmd". It returns a list with the R version of the arguments with a left parentheisis.

Value

a list of functions to convert the arguments of a matlab function. It comes with the names of matlab functions.

Examples

```
funcMap <- makeFuncMaps("trace: sum, diag(%1)")
funcMap[['trace']]$argMap[[1]]("matThing")
#$rargs
# "sum(diag(matThing)"

funcMap <- makeFuncMaps("mod: , 1 %% 2")
funcMap[['mod']]$argMap[[1]](c(4, 2))
#$rargs
# "(4, %%, 2"

test1 <- "mat"
test2 <- c("mat", "2")

funcMap <- makeFuncMaps(c("size--if 1:dim, 1", "size--if 2: ,dim(%1)[%2]"))
rightConv <- funcMap$size$flags$multSwitch(test1)
funcMap$size$argMap[[rightConv]](test1)
```

```
#$rargs
 "dim(mat"

rightConv <- funcMap$size$flags$multSwitch(test2)
funcMap$size$argMap[[rightConv]](test2)
#$rargs
 "dim(mat)[2]"
```

`makeSliceMap` *Make the maps for converting slice notation*

Description

Make the maps for converting slice notation

Usage

```
makeSliceMap(leftSym, rightSym, rClass, matClass = "")
```

Arguments

`leftSym`	The left symbol that contains the Matlab data
`rightSym`	the right symbol that contains the Matlab data
`rClass`	The formal r class name that defines what the R data is outputted as
`matClass`	The name of the Matlab class that should be converted

Details

Slice notation for matrices are tricky because they can easily be confused with the requirements for conversion are the bounds given by both left and right symbols or the Matlab class. The Matlab class allows for the conversion of structures but is really just a dictionary for the different bounds.

Value

A function that takes in a string and converts all the given slice notation

Examples

```
sliceMap <- makeSliceMap("{", "}", "list")
sliceMap("junk <- importData{300}")
# "junk <- importData[[300]]"

sliceMap <- makeSliceMap(matClass = "structure", rClass = "list")
sliceMap("junk <- students.AP.GPA")
# junk <- students[['AP']][['GPA']]
```

mat2r *mat2r*

Description

The top level driver function to call the converting functions and handle the input and output.

Usage

```
mat2r(
  inMat,
  pathOutR = "",
  funcConverters = NULL,
  dataConverters = NULL,
  verbose = 1
)
```

Arguments

inMat	A file path with the input Matlab / Octave code to be converted or a character vector of the code that needs to be converted
pathOutR	A file path with the desired output file location
funcConverters	A list of function converters that the user wants to use in this conversion made by makeFuncMaps
dataConverters	A list of data converters that the user wants to use in this conversion made by makeSliceMap or makeDataMap
verbose	A number indicating the amount of messages that should be outputed. **0** No messages **1** A summary report of what happened in the conversion **2** The final code as a message as well as the summary report

Value

A list containing the original code (named matCode) and the converted code (named rCode).

Examples

```
matIn <- c("function [ dat ] = xlsReadPretty(varargin)",
 "  didThing = 1*3;",
 "  dat = didThing / 3;",
 "end")
 mat2r(matIn, verbose = 0)$rCode

# "xlsReadPretty <- function(...){"
# "\tvarargin <- list(...)"
# "  didThing <- 1*3"
```

```
# "  dat <- didThing / 3"
#"\treturn(dat)"
#"}"
```

---

matconv *matconv: A Utility to Convert Matlab / Octave Code into R Code*

Description

Transferring over a code base from Matlab to R is often a repetitive and inefficient use of time. This package provides a translator for Matlab / Octave code into R code. It does some syntax changes, but most of the heavy lifting is in the function changes since the languages are so similar. Options for different data structures and the functions that can be changed are given. The Matlab code should be mostly in adherence to the standard style guide but some effort has been made to accommodate different number of spaces and other small syntax issues. This will not make the code more R friendly and may not even run afterwards. However, the rudimentary syntax, base function and data structure conversion is done quickly so that the maintainer can focus on changes to the design structure.

Index